TECHNICAL UNIVERSITY OF MUNICH

TUM Data Innovation Lab

Process Mining and Natural Language

Authors	Claas Brüß, Eduardo Goulart Rocha
	Gayatri Kudchadker, Manisha Singhal, Rachid Ellouze
Mentor	M.Sc. David Becher, Celonis SE
Co-Mentor(s)	M.Sc. Philippe Suennen (TUM), M.Sc. Oleh Melnyk (TUM)
Project Lead	Dr. Ricardo Acevedo Cabra (Department of Mathematics)
Supervisor	Prof. Dr. Massimo Fornasier (Department of Mathematics)

Jul 2019

Abstract

Process Mining is quickly emerging as one of the dominant trends in business intelligence and compliance. By utilizing the digital footprints and event logs generated during commercial operation, the actual inherent processes can be monitored, visualized and understood enabling far more informed strategic decision making. Celonis SE is a Munich-based company offering an analytical framework that aims to provide a process focused view of business data stored in ERP systems such as SAP through intuitive visual dashboard interfaces. The dashboards are customized or created from scratch by Celonis trained analysts on the client side, offering a tailored and ergonomic way to inform management decisions.

This approach has the advantage that only a few users (the analysts) need to go through time-consuming training in analysis creation. But as a consequence, users of the platform dependend on the Analyst to built their analyses. The goal of this project is to make relevant insights even more accessible to decision makers by offering a range of Celonis framework features through an user friendly natural language interface. In addition, unusual trends or changes in operation are highlighted to draw the attention of management to problem areas of high importance.

The project is split into two parts: Firstly a chat-bot that can retrieve data to answer natural language queries is built. Secondly, anomaly detection algorithms are applied to specified data subsets to identify possible outliers causing bottlenecks in the process.

The assistant can also be used to take multiple actions within Celonis and to trigger routines, making process mining more iterative.

Contents

1 Introduction

Today, nearly every process within modern companies is supported, controlled or documented by digital systems. Every digital operation produces valuable information in the form of log files, that for decades were mostly used for debugging and recovery. **Process mining** is the act of deriving insights from and improving processes through the aggregation and analysis of event log data. The Celonis framework offers a real time "snapshot" of a company and can in turn be used to increase transparency and compliance as well as to reduce cost by avoiding bottlenecks.

1.1 Cases, Activities and the Process Graph

For this project, the Purchase-to-Pay (P2P) model of an undisclosed example company was considered. The **Data Model** specifies how the SAP standard tables, acting as data sources, are connected through foreign keys. The data model for our particular case is visualized in Figure 1. Like every analyses at Celonis, the **Activity Table** is present (_CEL_P2P_ACTIVITIES). This table always contains the **Case ID** denoted by "_CASE_KEY", the **Activity Label** in this case "ACTIVITY_EN" and a **Timestamp** (" EVENTTIME").

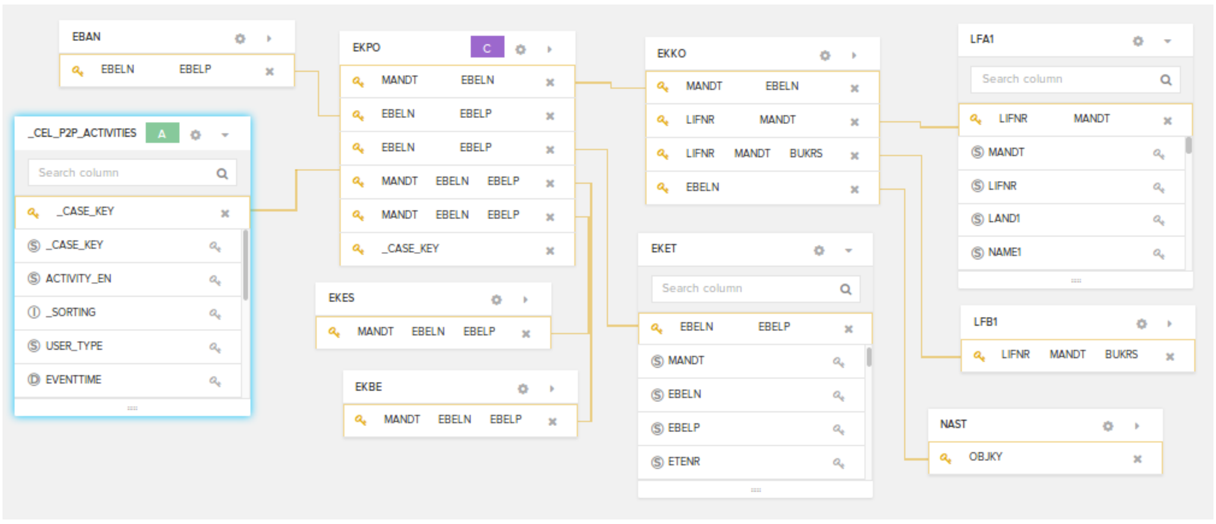
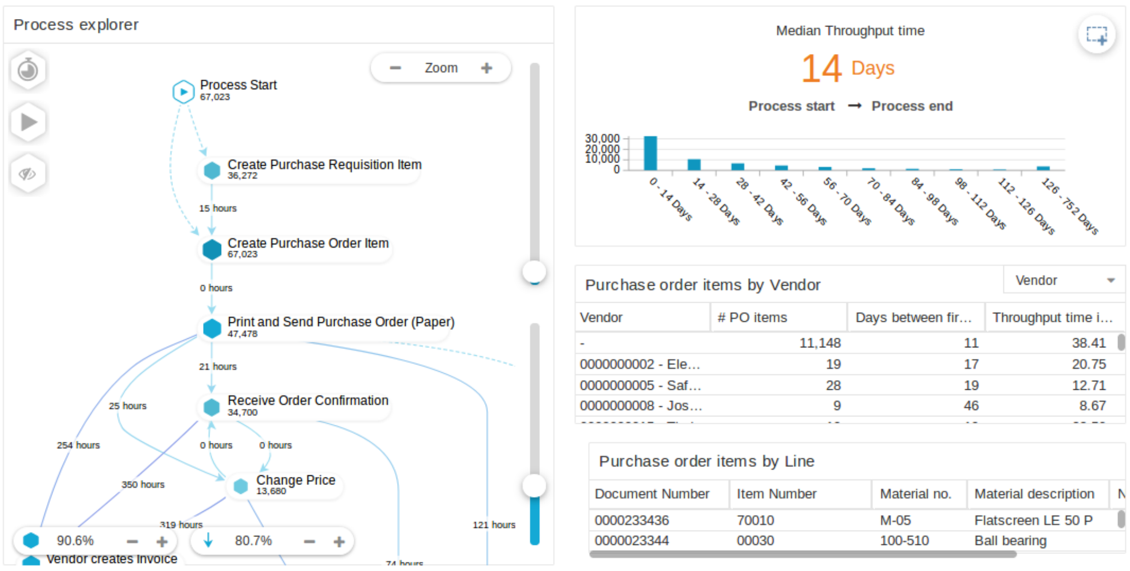


Figure 1: Purchase-to-Pay Schema

In Process Mining, a **case** refers to an instance of a process. In the P2P example, this corresponds to one **Purchase Order Item**. Individual stages in the process are called **Activities**, with examples including "Create Purchase Order Item" or "Change Price". The timestamps are used to order the activities so that a graph displaying process paths as shown in figure 2 can be built. This allows for further analysis of the data in order to answer business questions such as "How many cases took more than 15 days to finish?".



Purchase order items by Vendor

Vendor

Vendor	# PO items	Days between fir…	Throughput time i…
-	11,148	11	38.41
0000000002 - Ele…	19	17	20.75
0000000005 - Saf…	28	19	12.71
0000000008 - Jos…	9	46	8.67

Purchase order items by Line

Document Number	Item Number	Material no.	Material description
0000233436	70010	M-05	Flatscreen LE 50 P
0000023344	00030	100-510	Ball bearing

Figure 2: Celonis Process Analytics

Celonis splits users in 3 roles:

- The *Data Scientist* is responsible for building the Data Models, managing imports of data, installing and maintaining the software.
- The *Analyst* builds analysis sheets similar to figure 2, which are used by the Viewer.
- The *Viewer* is anyone that interacts with the analysis to answer business questions.

The idea behind this separation of roles is that viewers (usually decision makers) won't need any deep knowledge of the Celonis framework to analyze their processes. Instead, they refer to analyses built by the Analysts. However, to access information which is not directly displayed at the analysis, the viewer has to either dig further into the analysis, by applying a set of filters or ask the analyst to edit the analysis and include a component that displays it. Both alternatives are very time-consuming. Furthermore, the analysis only displays the data. Making sense out of this data still is the task of the Viewer and might not be simple due to the static design of analysis sheets. A flexible way of accessing data in Celonis is through its **Process Query Language**, which is discussed in the following section.

1.2 Process Query Language

Celonis develops its own query engine, with its own query language, named **Process Query Language (PQL)** . The syntax of PQL is closely related to the SQL standard, but simpler and specialized for process related queries. In contrast to SQL, one doesn't need to define how tables are joined. Tables are joined implicitly from the selected columns using the foreign keys relations specified in the data model. Another important difference is that sub-queries are not supported in PQL.

Listing 1 shows an example of a query in PQL which consists of a set of filters, a TABLE statement, which might contain an aggregate or not, and clauses to limit and order the data. Note that "LFA1"."NAME1" and "EKPO"."NETWR" are located on different tables, but the framework is able to find the join path from one table to the other based on the foreign keys defined in the Data Model. Furthermore, there is no "Group by" clause. Instead, data is grouped implicitly based on the selected columns. Although PQL is simpler than SQL, learning it still requires some time, specially for non-technical users. Therefore, a solution consiliating the flexibility of PQL with the ease of use of the analysis dashboards would be beneficial.

```
FILTER ''EKPO''.''NETWR'' > 10;
TABLE(SUM(''EKPO''.''NETWR'') AS COL1, ''LFA1''.''NAME1'' AS NAME)
LIMIT 10 ORDER BY NAME DESC
```

Listing 1: Example of PQL query which selects the top 10 vendors ("LFA1"."NAME1") and the sum of their net order values ("EKPO"."NETWR") where "EKPO"."NETWR" is greater than 10, ordered by vendor name.

1.3 Problem definition and goals of the project

The goal of this project is to develop an intelligent assistant that automates various tasks that usually fall under the responsibility of an analyst in Celonis. In addition, the assistant should be able to suggest metrics to look at, such as anomalies and process aspects with high improvement potential. It should also provide a user friendly and intuitive interface that can provide access to the desired output without the need for the decision maker to have any prior knowledge on how to use the front-end. This would take workload of analysts and reduce the waiting time for decision makers with the anomaly detection drawing attention to problematic patterns that no one might have previously considered.

The Natural Language Processing (NLP) Assistant and the anomaly detection approach were developed in parallel and in close coordination to be finally combined into a singular solution.

2 Natural Language Assistant

Recent advances in NLP have made chatbots a viable interface for humans to interact with digital systems. Natural language interfaces are easy to use and require no previous knowledge from the user. The goal of this part of the project is to add a natural language interface that helps the user mine the process in a more intuitive way. Furthermore, a chatbot might be a way of accessing data which is not directly displayed in the Analysis.

2.1 Intent based Assistant

The first approach we took was an intent-based approach. The Analyst can define a set of **intents** which are triggered through **utterances**. Intents should be seen as an action to be executed by the bot. When the user talks with the chatbot, it detects if the query corresponds to one of the predefined intents and if yes, the bot executes the action associated to it.

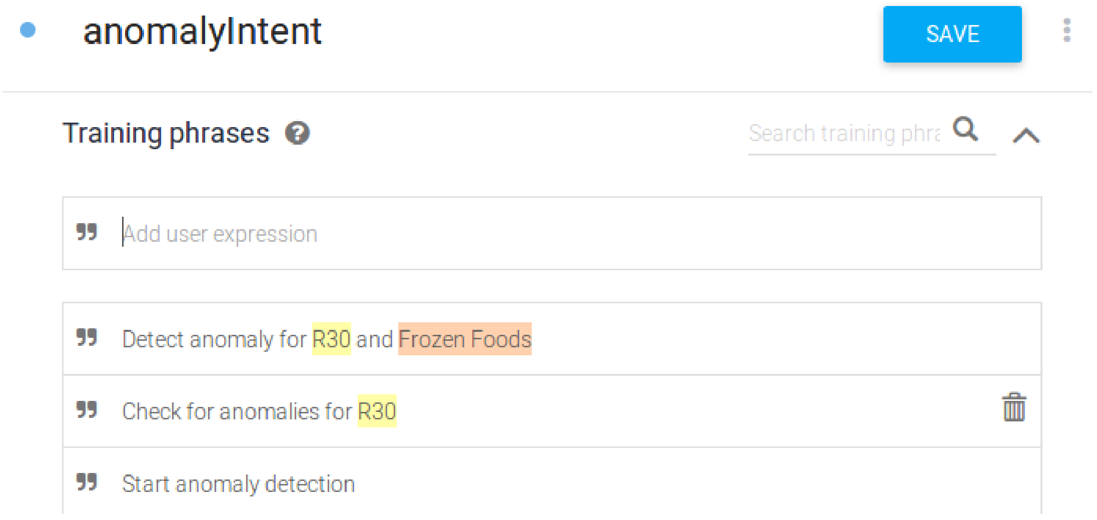
Intent recognition is a widely researched area in NLP and has reached maturity to a point where it's offered as a service. This makes world class **Intent Recognition** algorithms available to everyone. For this project, Google's Dialogflow was chosen as it offers a REST API to create new intents, making it more flexible. In Google Dialogflow, intents are created by providing a corresponding set of utterances to trigger the intent and parameters to be extracted. Domain-Specific Entity-types can also be defined by the user. Figure 3 shows the interface to add new intents and entities.

After adding a few intents, it became clear that an intent-based approach is too limited for 2 main reasons:

1. With too many intents, the algorithm which is usually based on measuring the similarity of example utterances and user's input, is unable to properly differentiate between intents. This is particularly true since the vocabulary used is rather limited, so two slightly different phrases might trigger two completely different intents. As an example, consider the utterances "sum of net order value is larger than 10" and "sum where net order value is larger than 10". Both differ by a single word and have different meaning, but since the only difference is in stop words, the bot might not understand the difference between them.

2. Maintaining the bot might be too complicated since the analyst would have to come up with all possible queries a viewer would ask. This creates a problem since we want the viewer to be able to answer questions which weren't previously defined by the analyst.

Due to the limitations of intent recognition, we decided to only use intents for triggering sequences of actions which can't be solved using a simple PQL query. For instance, to trigger an anomaly detection algorithm or some daily summary of the data. For questions like "How many cases do I have?", which resolve to a PQL query, we automatically build the query from the text.



anomalyIntent

Training phrases

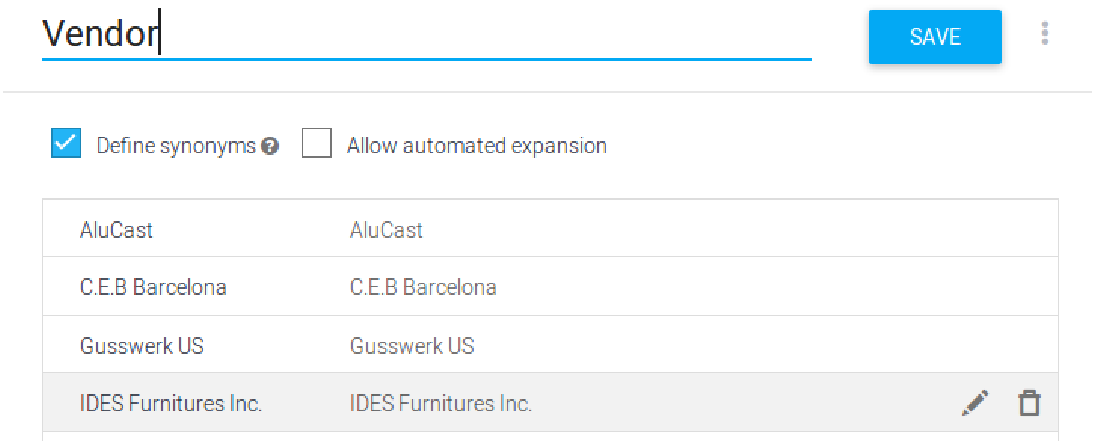
Add user expression

Detect anomaly for R30 and Frozen Foods

Check for anomalies for R30

Start anomaly detection

(a) Providing set of utterances that trigger "anomalyIntent". "R30" and "Frozen Foods" are user defined entities



Vendor

Define synonyms [✓] Allow automated expansion [ ]

AluCast	AluCast
C.E.B Barcelona	C.E.B Barcelona
Gusswerk US	Gusswerk US
IDES Furnitures Inc.	IDES Furnitures Inc.

(b) Defining "Vendor" entity

Figure 3: Google Dialogflow's interface to create new intents.

2.2 Automated sequence to PQL

Automated sequence to PQL is related to the task of *Semantic parsing*, which is a widely researched field in NLP. While there has been a lot of progress in the area, it hasn't reached a state where it can be considered as a solved problem. The challenge, as in all NLP tasks, lies in parsing an extremely ambiguous and context-free language to a formal language. Below we present a common approach to semantic parsing which served as an inspiration to our approach.

2.2.1 CCG- Semantic Composionality

The basic building blocks of **Combinatory Categorial Grammars (CCG)** [1] are the so called **categories**, which are composed of a syntactic and a semantic part. The syntactic part is composed of primitive symbols such as NP, ADJ and S and combination operators (\and /). The semantic part is a lambda DCS expression [2], which captures the action being expressed by the category. **Lexical entries** map a word or phrase to a category. The set of all lexical entries form a **lexicon**, as displayed in Figure 4a. The algorithm uses the lexicon to match words and phrases with their categories and apply a set of combination rules on the categories to derive the final form, as displayed in figure 4b.

Usually, at each step multiple combination rules can be applied, resulting in an exponential number of possible parsings. The algorithm must be able to select the correct parsing from all the possibilities. Artzi and Zettlemoyer [3] describes a weakly supervised algorithm to learn an heuristic to pick the best parsing from a set of alternatives by using some annotated data (pairs of utterances and the query answer) and an execution engine that evaluates the generated lambda expression. This has the advantage that one doesn't need to write the equivalent lambda expression for each query. However, defining the grammar still requires expert knowledge and is time consuming, which is the main reason why we didn't use CCGs in our project.

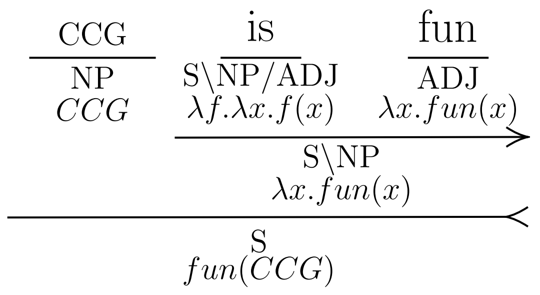
Instead, we can exploit PQL's simplicity in our favor. As mentioned before, PQL doesn't have join statements and sub-queries. Besides, the select statement for a chatbot can only have an aggregation function, since it is not reasonable to return full tables back to the Bot. The task reduces then to extracting the filters, the column and the aggregate that the user is interested at. In the next chapter, we present our implementation of semantic parsing.

$$\text{CCG} \vdash NP : CCG$$

$$\text{is} \vdash (S\backslash NP)/ADJ : \;\lambda f.\lambda x.f(x)$$

$$\text{fun} \vdash ADJ : \lambda x.fun(x)$$

(a) CCG Lexicons



(b) Parsing with CCG, adapted from ref()

Figure 4: Parsing with CCG. Left and Right arrows are combination rules. Other rules are possible. Adapted from [4]

2.3 Implementation

The implementation relies on spaCy, a free, open-source Python NLP library which offers tokenization, named entity recognition and rule based matching functionalities out of the box. The assistant interacts with Celonis through its Python API.

The assistant is designed to be model-independent and to work with as few external configurations as possible. Once the assistant connects to a Celonis Data Model, it can get all Key Performance Indicators (KPIs) which were configured for that analysis. Furthermore, Celonis provides a mapping file, that maps the technical name of the columns to human readable names. The bot uses information in this file to detect the columns in a query. This also ensures that the assistant is always up to date with the analysis. If new columns are added to the Data Model or if new KPIs are added to an analysis, the assistant can automatically add it to it's capabilities.

2.3.1 Parsing Pipeline

Our approach relies on spaCy's rule based Matcher [5]. One can define entities, lemmas and patterns. The Matcher passes through the text and searches for pattern matches. Pattern matches are emitted to the query object, which can also handle ambiguity. Figure 5 depicts the different steps of the pipeline for the query "The average throughput time for vendor SCT Inc.".

As already mentioned, the role of the parser is to map a sentence to a PQL query. For this, the following steps are performed:

1. **Tokenization**; first the text is segmented into words and punctuations. Tokenization is based on several rules that are language specific. For instance the points in the word "U.K." are part of the word and not for ending a sentence.

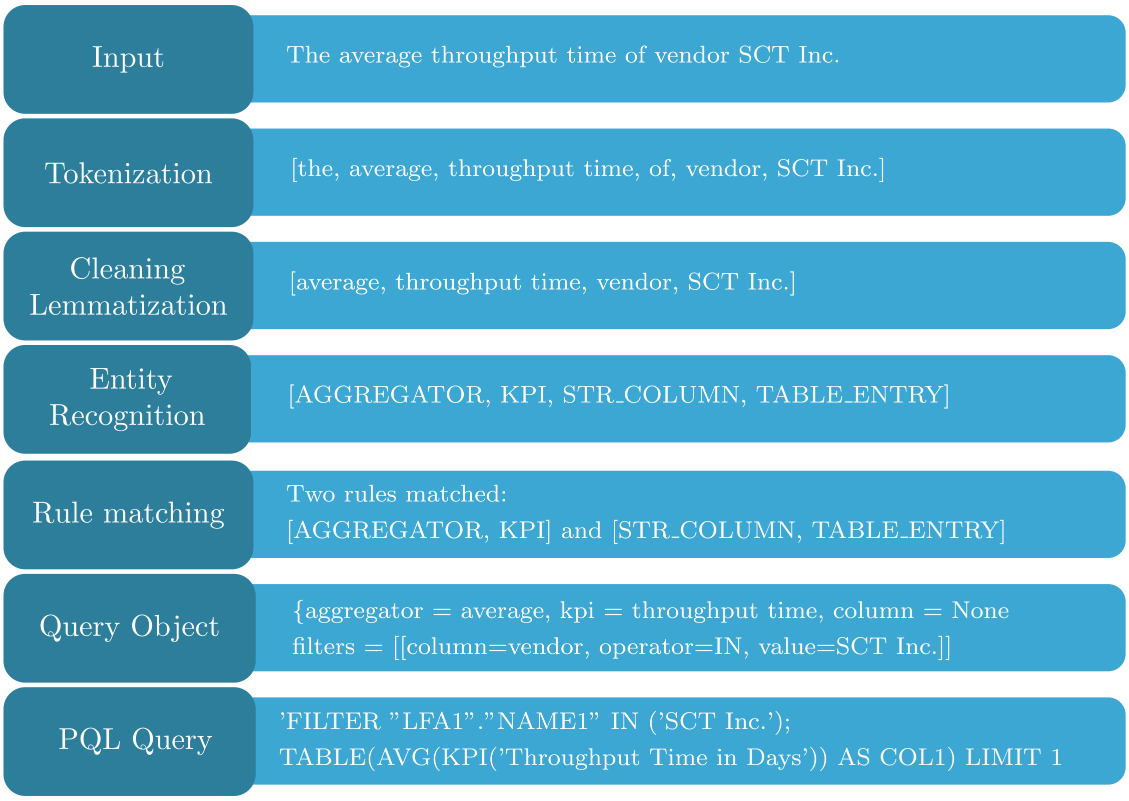
2. **Text cleaning and lemmatization**; here stop words such as "the", "by" or "so" are removed. Furthermore, words are replaced by their lemmas. For example, "is", "are", "were" are replaced with "be". Some stop words were kept if they have some special meaning, for instance 'A', which denotes a user type in SAP's P2P schema.

3. **Named Entity Recognition**; in this step, relevant entities are extracted from the input text. We distinguish between five different entities. The entities are the following (i) *AGGREGATOR*, (ii) *COLUMN*, (iii) *KPI*, (iv) *OPERATOR*, (v) *TABLE_ENTRY*. Note that we distinguish between columns with string entries and columns with numerical entries. The same applies for operators, we distinguish between numerical operators such as"equals" or "greater than" and logical operators such as "and" or "or".

4. **Rule based matching**; on top of the extracted entities we define a set of rules that help to parse the query. A rule is a pattern that consists of a sequence of entities, lemmas, POS tag, exact text, etc. Every time a rule is matched in the input text, a callback function is called. For better understanding, consider the rule "[*COLUMN*, *OPERATOR*, *NUMBER*]" and the input text "Average throughput time where the net order value is larger than 10". The *Matcher* will match the rule with the text "net order value is larger than 10" and call the appropriate callback function.

5. **Building the Query Object**; as already mentioned, when some patterns are matched, callbacks that modify the Query Object are triggered. The Query Objects can also work to handle context, resolve ambiguities or do some post-processing. Introducing it, facilitates the generation of the PQL query and embedding the result in a text.

6. **Building the PQL query**; after constructing the Query Object, building the query becomes easy, in which the syntax of PQL is taken into consideration.



Input	The average throughput time of vendor SCT Inc.
Tokenization	[the, average, throughput time, of, vendor, SCT Inc.]
Cleaning Lemmatization	[average, throughput time, vendor, SCT Inc.]
Entity Recognition	[AGGREGATOR, KPI, STR_COLUMN, TABLE_ENTRY]
Rule matching	Two rules matched: [AGGREGATOR, KPI] and [STR_COLUMN, TABLE_ENTRY]
Query Object	{aggregator = average, kpi = throughput time, column = None filters = [[column=vendor, operator=IN, value=SCT Inc.]]
PQL Query	'FILTER "LFA1"."NAME1" IN ('SCT Inc.'); TABLE(AVG(KPI('Throughput Time in Days')) AS COL1) LIMIT 1

Figure 5: Step by step parsing of sentence "The average throughput time for vendor SCT Inc." illustrating the parsing pipeline

2.3.2 Features

The assistant supports the following features:

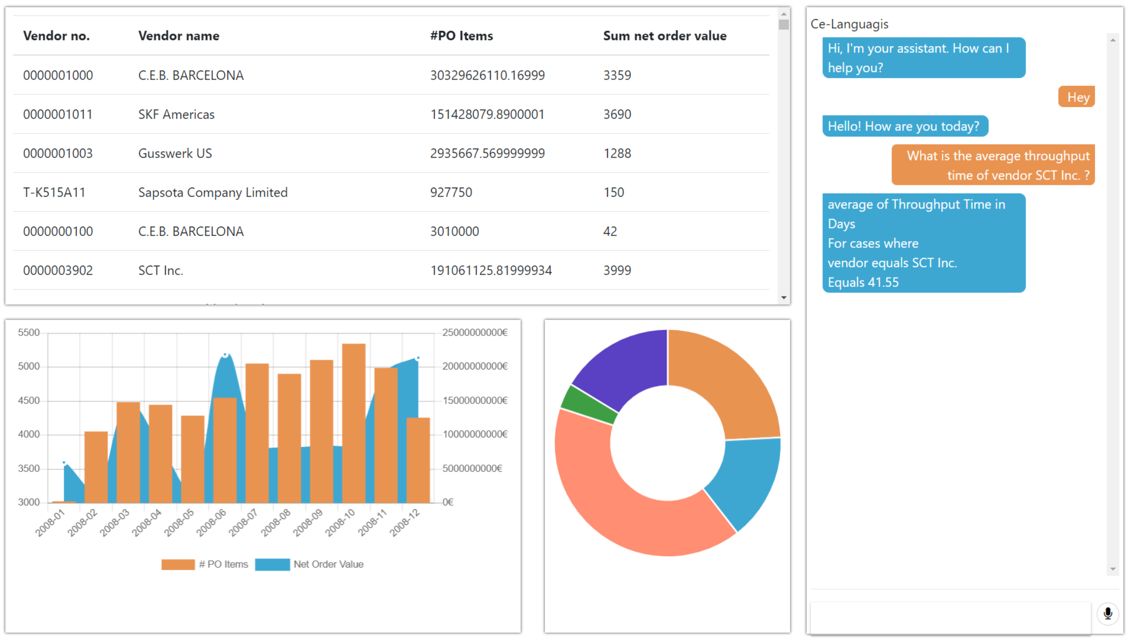
- **Speech recognition**; using Google-API [6], the assistant is able to translate speech to text.
- **Charts**; alongside with the chat-bot, the assistant provides build-in charts that represent data in form of a table, pie chart and histograms.
- **Auto-complete**; in order to provide a better user experience (UX), the assistant provides a smart auto-complete feature. The auto complete module is context based, in the sense that it analyzes the so far typed text to supply consistent suggestions. This also improves accuracy of the bot by helping the user to provide more structured queries.
- **Simple small talk**; the assistant is able to lead simple small talks, such as "Hello, how are you?".
- **Applying filters to update charts**; the users have the possibility to apply filters to the charts by simply typing the desired filter in a natural language text. For instance, typing "net order value larger than 10" to the chat-bot results in filtering the charts to display only data that satisfy this condition.

- **Parsing aggregator, KPIs and Columns**; the assistant is able to parse text input that contains an aggregator, a KPI or a Column, and multiple filters to a PQL query. It then runs the query and displays the result. For example, it constructs from the input "Average net order value of the vendor SMP", the query, `‘FILTER ‘‘LFA1’’.‘‘NAME1’’ IN (‘SMP’); TABLE(AVG(‘‘EKPO’’.‘‘NETWR’’) AS COL1) LIMIT 1’`.

- **Multiple filters and filter operators**; the parser is able to extract multiple filters from the input text. It also supports different operators for filtering, for instance, "equals", "larger than" or "in". Note that, we distinguish between string operators and numerical operators. Moreover, Celonis provides a powerful operator, namely the "like" operator. It offers the possibility to filter on a pattern or an expression. This functionality is also supported and can be extracted from the input text. Furthermore, the parser is able to apply logical operation on top of the filters, namely "and" and "or" operators.

- **PQL to sequence**; in the same frame of UX, the chat-bot embeds the returned result in meaningful text. It formulates the executed query in a human readable text format. This functionality provides feedback of how the assistant interpreted the question and thus it can be used to check the correctness of the result.

- **Detect hints that refers to anomaly**; the assistant is able to answer questions related to anomaly and unusual events such as "What was unusual last month?". More details about anomaly detection are discussed in Chapter 3.



Vendor no.	Vendor name	#PO Items	Sum net order value
0000001000	C.E.B. BARCELONA	30329626110.16999	3359
0000001011	SKF Americas	151428079.8900001	3690
0000001003	Gusswerk US	2935667.569999999	1288
T-K515A11	Sapsota Company Limited	927750	150
0000000100	C.E.B. BARCELONA	3010000	42
0000003902	SCT Inc.	191061125.81999934	3999

Ce-Languagis

Hi, I'm your assistant. How can I help you?

Hey

Hello! How are you today?

What is the average throughput time of vendor SCT Inc. ?

average of Throughput Time in Days
For cases where
vendor equals SCT Inc.
Equals 41.55

Figure 6: Assistant Interface

2.4 Results

The assistant is able to handle different types of queries and to apply complex as well as advanced filters to the charts. It can handle ambiguities in the text and ask back questions. Furthermore, it reformulates the question in the way it understood, in order to provide a feedback about the correctness of the results. It can adapt to different data models with little external intervention. The combination of the intent based recognition offered by DialogFlow with the automated query generation provides several benefits. On one hand, versatility of possible actions to be triggered, like an anomaly detection algorithm or a daily summary of the data, offered by intent recognition and, on the other hand, robustness offered by the automated query generation.

Conversation flows in a natural and uncomplicated way as displayed in 6, reducing the amount of knowledge needed to interact with the platform.

3 Anomaly Detection

In order to support strategic decisions we would like to draw attention to problematic patterns found through algorithmic analysis. In most cases analysts need to anticipate potential issues and problematic patterns to compose an analysis within the Celonis framework. To a certain extent, we aim to elevate the need for this kind of foresight by automatically detecting unusual process or non-conformant cases in order to draw attention of the decision makers to issues that might not have been addressed otherwise because nobody thought of looking for this specific non apparent pattern. In this scenario we consider **Anomalies** to be cases that are set apart either by a substantial difference in KPIs and metrics or a clear deviation from the predefined process flow.

The second task of this project was to create an intelligent mining assistant that can make proactive suggestions and direct the users to problems in their processes. Anomaly detection can help to improve process mining by detecting activities that are taking more than the required time or an activity that is repeatedly failing. Finding such processes and rectifying them will result in increased efficiency. Also, not all users are familiar with Celonis framework and it is not always possible to directly find anomalies when applying certain filters in the process explorer or other analytic views in the interface. The system should be proficient enough to analyze the case data and inform user about the anomaly if any.

3.1 Dataset

The provided data stems from SAP data entries describing a Purchase-to-Pay process. Parts of the data were fabricated to increase the data volume and we do not have clear indications concerning the data quality. The analyzed dataset encompasses roughly 67000 cases. Even though each case as a data entry is specified by over 200 columns combining data from various SAP standard tables, most columns are sparsely populated and only certain columns can be meaningfully applied as KPIs (e.g.: throughput time).

From an activity perspective, the process data lists 24 different activities and 347 unique connections between them, demonstrating a large variation in the process. The preconceived process path, dubbed the **Happy Path**, includes 7 activities with 6 connections between them.
For the purpose of analysis, the dataset can conceptually be split into 3 categories of data dimensions:

- Case Specification: These columns enumerate the case properties which are set from the process start. Included are such stats as "Vendor" or "Order Quantity".
- Case Metrics and KPIs: These are measured quantities, such as "Throughput Time" and status indicators which are not predetermined and are acquired during operation or after process completion.
- Process Flow Data: This data subset encodes the topological information of the case specific process flow. This is done by specifying connections as directed edges

with the start and end activity including the respective time stamps.

In our approach, the different data categories are utilized very differently, thus warranting such a distinctive categorization of columns.

3.2 Related Work

Lamba et al. [7] use a model-based approach to cluster sequences of user behaviours and find anomalous sequences. The authors model clusters using Markov chains and define a likelihood based statistic to score anomalous sequences. The paper exploited the contextual information of the technical architecture and was found to be a scalable and robust framework for detecting anomalies. Agrawal and Agrawal [8] present a survey on different data mining techniques for anomaly detection ranging from clustering-based techniques to neural networks. Silva and R Ferreira [9] describe a Hidden Markov Model(HMM) based Sequence Clustering algorithm for detecting unusual event log traces for process mining. They experiment with different ways of using HMMs in handling noise to generate log traces.

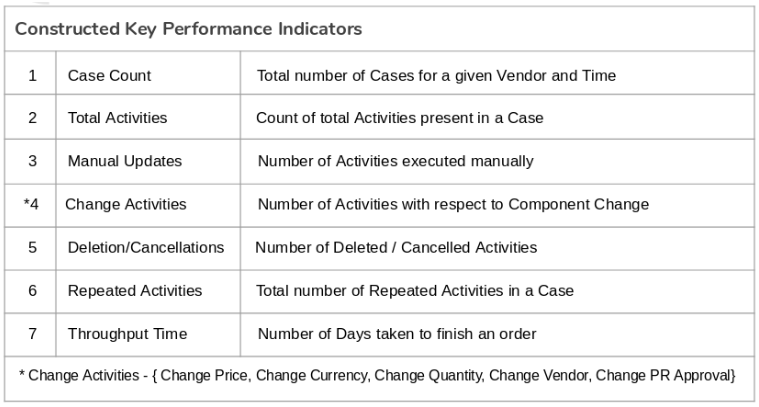
3.3 Approaches and Issues

The section describes the approaches that were considered for anomaly detection and other issues with the dataset in general.

- **Dataset Limitations**
  Without clear indications of data quality, very limited domain knowledge and partially fabricated data, we face multiple factors that aggravate the interpretation of result and thus hinder iterative improvement and tuning of methods. This on multiple occasions led to system inherent behaviors to be classified as anomalies during the first milestones of this project.

- **Markov Chains**
  Markov chains have been successfully used for anomaly detection in sequences. We wanted to apply the Markov model based clustering method as described by [7]. The paper proposes a method where a set of clusters are modelled as Markov chains by initializing with random transition and initial probabilities. The optimal set of clusters is found using Bayesian Information Criteria. The probabilities per each cluster are updated based on training example which belongs to the cluster. Each training sequence is assigned a score based on it's likelihood of belonging to a particular cluster. If the score falls below a threshold, then the sequence is marked as anomalous. This approach marks a sequence as anomalous if a specific sequence or the most likely sequence is not being followed by a training example. With respect to the business process data, apart from sequences which do not follow an ideal path, sequences that follow a desired path but have an extreme value for a KPI e.g. a high throughput time, should also be reported to the user. Using the approach described in the paper will not classify such sequences as anomalies since they follow a desired path.

- **Clustering**
  Clustering is an unsupervised learning algorithm which separates a set of items into groups such that within-group similarity is higher than between-group similarity. Clustering helps to identify a set of outliers which have striking differences from the rest of the data points, and these outliers are termed as anomalies. We used K-means clustering to find deviations in the P2P model. Instead of using all available features from the dataset, we identified 'Vendor' as a high priority dimension based on it's importance in Celonis analysis. Data was filtered on a specific Vendor. This can be done because it can be safely concluded that different Vendors follow a different set of steps for processing orders. So it is unreasonable to cluster without pre-filtering. Filtering allows us to do more accurate prediction. We constructed a set of KPIs as presented in Figure 7. Once filtered, the constructed KPIs were provided as feature vectors to cluster the dataset.



Constructed Key Performance Indicators		
1	Case Count	Total number of Cases for a given Vendor and Time
2	Total Activities	Count of total Activities present in a Case
3	Manual Updates	Number of Activities executed manually
*4	Change Activities	Number of Activities with respect to Component Change
5	Deletion/Cancellations	Number of Deleted / Cancelled Activities
6	Repeated Activities	Total number of Repeated Activities in a Case
7	Throughput Time	Number of Days taken to finish an order
* Change Activities - { Change Price, Change Currency, Change Quantity, Change Vendor, Change PR Approval}		

Figure 7: KPIs

The elbow method was used to determine optimal number of clusters. For multiple Vendors, 7 was the ideal number of clusters. The clusters obtained were showed similarities between a set of KPIs. But we could not identify a cluster which showed deviations with respect to all/maximum set of KPIs and clearly deviates from the group. Also, for some Vendors the elbow plot did not appear to converge to a particular number but decreased with more number of clusters. Thus, we concluded that K-means clustering was not a suitable approach.

3.4 Implemented Approach

Our final approach uses simple statistics to find anomalies. We have replicated and automated the Celonis Anomaly Analysis. Cases were split into sub-sets based on case specification data, i.e. 'Vendor'. Figure 8 shows the worklow of the approach. Due to the limited time frame of the project we focused on metrics-based analysis. Path-based and topological features analyses were not implemented due to time constraints and can be addressed as future work.

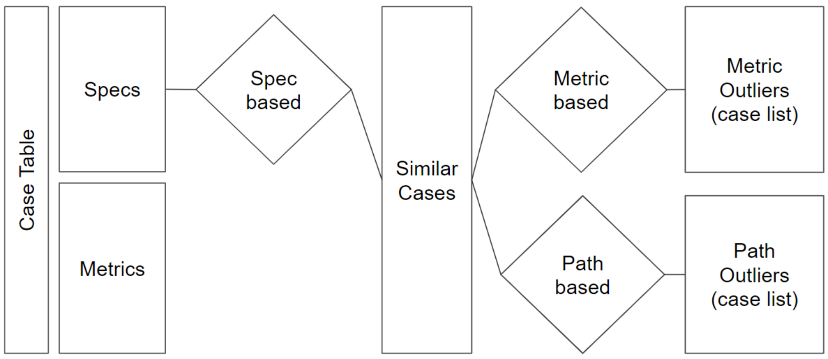
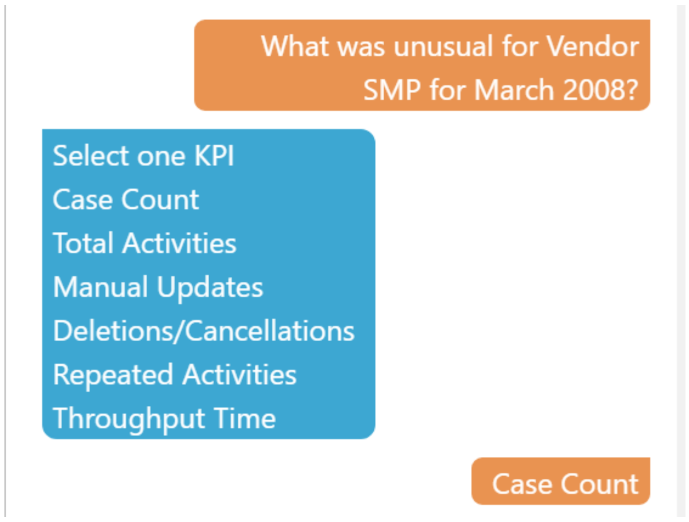


Figure 8: Filtering Cases based on spec similarity

The implemented approach requires 'Vendor' and 'Event Timespan' as mandatory input dimensions. i.e. when a user asks a query, it is necessary to provide the Vendor name and time period for which the user will like to have the information. Material and Purchasing Group are optional dimensions. The timespan should specify if the user wants the details per Year/Month/Week and respective dates.

The user is asked to select a KPI from the list of KPIs presented in Table[7]. These KPIs are hard-coded in the script, but it is possible to add new key indicators by modifying the PQL query to incorporate additional KPIs and adding conditions to process the new KPIs in the existing Python script. Figure 9 shows the dialog flow displayed in the assistant.



What was unusual for Vendor SMP for March 2008?

Select one KPI
Case Count
Total Activities
Manual Updates
Deletions/Cancellations
Repeated Activities
Throughput Time

Case Count

Figure 9: Anomaly Dialog Flow

Anomalies are reported per activity level for 'Change Activities', 'Manual Updates', 'Deletions/Cancellations', 'Repeat Activities'. Activity level analysis evaluates individual activities for a given time period and reports changes in the activities. The data is grouped by timestamp and ratio of change activities, manual updates, deletions, repetitions is found with respect to total activities. The assistant reports a percentage change between the queried and past date when the vendor was operational to the user as an output, along with a graph presenting variation in KPIs over the past dates.

We assumed a threshold of 50% change to identify an anomalous KPI. If the percentage change in a KPI is greater than 50%, user is asked to verify the process for the queried time-span. The analysis also showed that most of vendors in the dataset have more than 10 activities per Week, more than 30 activities per Month and more than 600 activities per Year. Thus, percent change is found only if the median value of total activity for any vendor is at-least 5 activities per Week, 25 activities per Month or 500 activities per Year. If the median is less than the specified value, then absolute change in the exact value of the kpis is reported without finding the ratios.

Anomalies for 'Throughput Time (in Days)' are reported per Case level. Case level analysis evaluates individual cases and reports cases which deviate from the rest in the group. The data is grouped on Case IDs. Dataset analysis revealed that throughput times for most of the vendors is right skewed. Thus, we find the skewness for throughput time in days for a given vendor. For a right skewed distribution, the cases with throughput times greater than the 95th percentile are evaluated further. If the distribution is left skewed, the cases with throughput times lesser than the 5th percentile are processed further. It is important to consider cases with very low processing times, as it is possible that some cases are skipping a mandatory and a critical activity due to which the order gets finished quickly. We assume 95th and 5th percentile as a reasonable estimate which provides cases with extreme throughput days. For an unskewed distribution, the top five cases with maximum time duration are evaluated. We find the anomaly factor for these separated cases which is given as follows:

$$\begin{aligned}\text{Anomaly Factor} = \text{Change Activity Ratio} + \text{Manual Update Ratio} +\\ \text{Delete Cancel Ratio} + \text{Repeat Activities Ratio}\end{aligned}$$

The assistant presents a distribution graph of throughput time to the user with some explanation and five cases(if present) with their variants having high anomaly factor.

3.5 Results

The approach is implemented using Python. The Celonis Python API is used to generate queries for data retrieval. The following section explains the results returned for different variations of queries being asked by the user.

1. **Query:** What was unusual for Vendor AluCast for March 2009?
   KPI selected: Case Count
   Inputs: Vendor = AluCast, Span = Month, Date = 2009-03-01, KPI = Case Count
   Output: Result returned is displayed in Figure 10.
   Since the percentage increase in Case count is 80% which is more than the 50% threshold, assistant asks the user to verify the activities of the vendor for the queried month. If the percentage was less than threshold the assistant would report "Nothing unusual happened in the given time"

Timestamp	CaseCount	TotalActivity
2009-02-01	80	212
2009-03-01	144	597

```
Vendor: AluCast
Queried Month: March 2009
Last Month for which the vendor was operational: February 2009
There was a increase in Cases by 80.0%
Please verify why there is more than 50% increase in Cases in the queried Month.

<Figure size 1080x720 with 0 Axes>
```

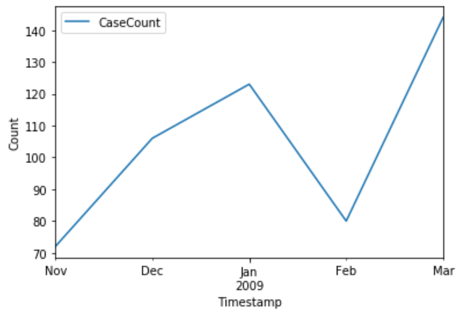


Figure 10: Output for Query 1

2. **Query:-** What was odd for Vendor Unisono AG for third week of March 2009 for the material 'Frozen Foods' and purchasing group 'R30'?
   KPI selected:- Throughput Time
   Inputs:- Vendor = Unisono AG, Span = Week, Date = 2009-03-16, KPI = Throughput Time, Material = Frozen Foods, Purchasing Group = R30
   Output:- Result returned is displayed in Figure 11.
   The output explains the graph and displays the anomaly factor calculation.

	CaseId	ThrDays	anomalyFactor	Variant
0	122080	120.0	0.5	Create Purchase Order Item, Print and Send Pur...

```
Vendor: Unisono AG
Material: Frozen foods
Purchasing Group: R30
The Throughput Time plot is skewed to the right.
The 95th percentile value is 116.0 which is 82.29 higher than the average.
The table shows cases with throughput time above the 95th percentile having highest anomaly factor.
Anomaly Factor = Change Activity Ratio + Manual Update Ratio + Delete Cancel Ratio + Repeat Activities Ratio
```

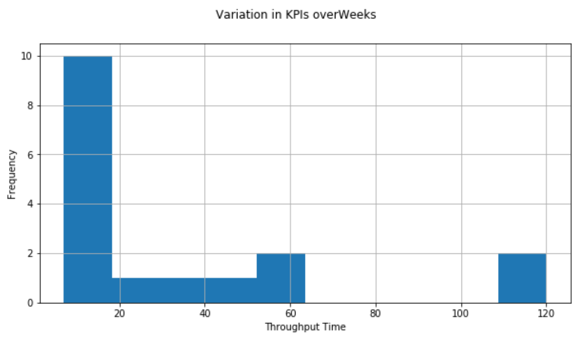


Figure 11: Output for Query 2

3.6 Future Work

The assistant at present detects anomaly in data only when Vendor is mentioned with the time period in which anomaly is to be observed. Also, it only consider one KPI that should be specified by the user. Though the assistant is smart enough to detect anomaly for specific KPIs, it can be made smarter so that it can detect anomaly without applying any filter and can consider necessary KPIs altogether.

One approach is to use network analysis that can observe all the paths between activities and find pattern in them. Taking metrics like throughout time as weights, it can be used to detect unusual paths and cases in which this pattern is recurrent and those cases can be further scrutinized with other KPIs like automation rate.

Second approach can be using machine learning algorithms like clustering and Markov chain methods to cluster cases taking different KPIs as weights and to find path-based anomalies. But this requires large and balanced dataset. The given dataset is not symmetric as some Vendors have very large amount of cases and for some there are only handful of cases so the clustering approach in the current dataset will be biased towards large vendors.

4 Summary

In this project, Celonis platform was enhanced by adding an Intelligent Assistant that helps users mine their processes in a more intuitive way. This assistant takes questions from user in natural language and translates it into PQL query to get the desired output.

The assistant can also be used to detect anomalous activities or cases. If users want to know the reason for the delay in their orders or why it got cancelled, they have three approaches: first, ask the analyst themselves and wait for their reply which might take a lot of time; second, use Celonis framework and apply filters to see if something anomalous occurred in their order. But this approach needs domain knowledge about how Celonis framework works and which filters to apply. Not all users know how to use the framework and even if they do, it is not guaranteed that the applied filters will show aberrant result. Third approach is to use this assistant to ask what unusual happened for the particular vendor in a given time period and in which KPI you are most interested in.

Given the short time-frame of this project, using annotated data to deal with query ambiguity was not possible. Celonis, as a company, has the means (man-hours) to invest in such annotation effort and we strongly believe that the assistant can reach production level and be deployed as a product with a bit more of perfectioning. Besides that, Celonis' platform can also provide more functionalities for the bot that would allow to answer more complex queries, for instance "Which vendors have a throughput time larger than average?", which translates to a complicated PQL query, but that would be simple if sub-queries were allowed. To be successfully integrated, the assistant shouldn't be seen as an application running on top of Celonis, instead it should become an application within Celonis, such that new features of the platform also take it into consideration.

Anomaly detection can also be improved if more data is given so that machine learning algorithms can be applied to get better accuracy. Also to detect abnormal behaviour of activity, good domain knowledge is required which can help in feature engineering and selecting attributes that interests clients.

Bibliography

[1] Susan Steele. “Mark Steedman,Surface structure and interpretation (Linguistic Inquiry Monographs 30). Cambridge, MA: MIT Press, 1996. Pp. xiv 126.” In: *Journal of Linguistics* 34.2 (1998), 489â“549. DOI: `10.1017/S0022226798317119`.

[2] Percy Liang. “Lambda Dependency-Based Compositional Semantics”. In: *CoRR* abs/1309.4408 (2013). arXiv: `1309.4408`. URL: `http://arxiv.org/abs/1309.4408`.

[3] Yoav Artzi and Luke Zettlemoyer. “Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions”. In: *Transactions of the Association for Computational Linguistics* 1 (Dec. 2013), pp. 49–62. DOI: `10.1162/tacl_a_00209`.

[4] Luke Zettlemoyer Yoav Artzi Nicholas FitzGerald. *Semantic Parsing with Combinatory Categorial Grammars.* 2014. URL: `https://yoavartzi.com/tutorial/` (visited on 07/27/2019).

[5] SpaCy. *Spacy Matcher.* `https://spacy.io/api/matcher/`. [Online; accessed 26-July-2019]. 2019.

[6] Google. *Google Speech-to-Text.* `https://cloud.google.com/speech-to-text/`. [Online; accessed 26-July-2019]. 2019.

[7] Hemank Lamba et al. “Model-based Cluster Analysis for Identifying Suspicious Activity Sequences in Software”. In: *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics, IWSPA@CODASPY 2017, Scottsdale, Arizona, USA, March 24, 2017.* 2017, pp. 17–22. DOI: `10.1145/3041008.3041014`. URL: `https://doi.org/10.1145/3041008.3041014`.

[8] Shikha Agrawal and Jitendra Agrawal. “Survey on Anomaly Detection using Data Mining Techniques”. In: *19th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2015, Singapore, 7-9 September 2015.* 2015, pp. 708–713. DOI: `10.1016/j.procs.2015.08.220`. URL: `https://doi.org/10.1016/j.procs.2015.08.220`.

[9] Gil Silva and Diogo R Ferreira. “Applying Hidden Markov Models to Process Mining”. In: Jan. 2009. DOI: `10.13140/2.1.4120.7689`.