Analysis of Information-Seeking Conversations with Process Mining

Alexander Holstrup
DTU Compute
Technical University of Denmark
Kgs. Lyngby, Denmark
abho@dtu.dk

Lasse Starklit
DTU Compute
Technical University of Denmark
Kgs. Lyngby, Denmark
s165498@student.dtu.dk

Andrea Burattin
DTU Compute
Technical University of Denmark
Kgs. Lyngby, Denmark
andbur@dtu.dk

***Abstract*—Online communities have become increasingly popular sources of information for both users and organisations. Every day thousands of users ask questions on these platforms, yet this knowledge-sharing process is not very studied. In this paper we aim to fill this knowledge-gap, by providing a general framework for studying the knowledge-sharing processes in such online communities. Specifically, we provide a three-step algorithm, that can create process models from interleaved and unlabelled conversations. We provide an instantiation of our framework, and conduct several experiments to evaluate its performance using the process mining tool Disco. From these experiments we show that it is possible to gain meaningful insights from the conversations on online communities using process mining techniques.**

***Index Terms*—Process Mining, Information-seeking Conversations, Classification**

I. Introduction

Over the last decades the increasing accessibility to the Internet has been a huge factor to the disruption of traditional letter communication. With the advancement of this technology, people can now obtain important information in a heartbeat, without need of heavy encyclopedia or other physical writings. The type of conversations in which a person is reaching out to another one to gather knowledge on a certain topic is referred to as *information-seeking conversations*. In the scope of this research, a message stream, is a series of messages, where the first message will be an information-seeking request which is typically eventually followed by a fulfillment of such request.

Today, an information-seeking utterance, is not only aimed at people, but other sources of information too. An emerging phenomenon within discussion platforms concerns the growing diffusion of chatbots. Statistics and predictions report[1] that, by 2020, 80% of enterprises will use chatbots and, by 2022, banks can automate up to 90% of their customer interaction using chatbots. Nowadays, these bots are typically used to answer simple common questions from users. However when users request information, which require multiple rounds of conversation (referred to as a multi turn conversation), the chatbots are lacking structural sense[2].

[1]See, for example, https://chatbotsmagazine.com/chatbot-a487afec05b (accessed in Jan 2020), where statistics about chatbots diffusion and challenges are reported.

Today, an information-seeking utterance, is not only aimed at people, but other sources of information too. An emerging phenomenon within discussion platforms concerns the growing diffusion of chatbots. Statistics and predictions report[2] that, by 2020, 80% of enterprises will use chatbots and, by 2022, banks can automate up to 90% of their customer interaction using chatbots. Nowadays, these bots are typically used to answer simple common questions from users. However when users request information, which require multiple rounds of conversation (usually referred to as a multi turn conversation), the chatbots are lacking structural sense[2]. Conversational process mining, has the potential to give these chatbots structural sense, and improve their capabilities. We envision, that the framework outlined in this paper, will aid the future development of chatbots, as well as future research of conversational process mining.

Our research reveals how the structure varies between online communities. Understanding and being able to uncover structural patterns of human-to-human conversations from different online communities, will reveal crucial information towards the improvement of communications with chatbots.

In this paper, we present a general framework, along with a corresponding instantiation, that enables analysts to extract process models from interleaved and unlabelled conversations observed in online communities using process mining (specifically, control-flow discovery) techniques [1].

The framework consists of three steps: the first is called *disentanglement* and it maps every message to its corresponding conceptual conversation. This is a necessary step since multiple conversations can be interleaved and progressing at the same time. The succeeding step performs *utterance classification*, to assign to each message a label referring to the purpose of the message itself. The last step of the framework associates a set of *topic-labels* to each message. The output of this framework, is a structured and labelled log of the conversation, where each message is mapped to its utterance, to the conversation it belongs to, and to the set of topics under discussion. Therefore, the log structurally enables the analysis with process mining techniques [1]. In this paper we instantiate

[2]See, for example, https://chatbotsmagazine.com/chatbot-a487afec05b (accessed in Apr 2020), where statistics about chatbots diffusion and challenges are reported.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

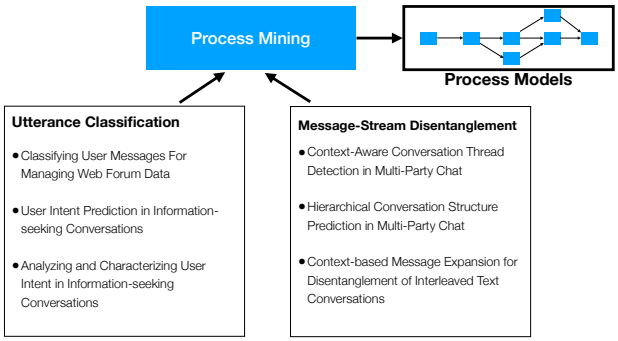


Fig. 1. The sub-problems required to perform process mining from conversational data.

the proposed framework, and use it on two real-world datasets, and a synthetic one, with the aim of proving its potential.

The rest of the paper is structured as follows. In Section II we introduce related work within utterance classification, message thread disentanglement and process mining of conversations. In Section III we describe our research questions and conceptually describe our proposed framework. In Section V we present our experiments and results. Section VI and VII present the discussion and conclusion.

II. Related Work

In this paper we present a process mining based framework for analyzing information-seeking conversations. In order to automatically analyze data with process mining it is necessary that each data point is structured according to a precise schema [1] (i.e., each data point has to contain, at least, the activity performed, the process instance the activity refers to, and a timestamp). Therefore, to automatically go from a conversation dataset (where each message contains at least user-id, timestamp and text of the message) to a process mining-capable one, some pre-processing is required. During this phase, we address the following two problems, cf. Fig. 1: the *message-stream disentanglement*, i.e., identification of conversation threads within a single stream of messages; and *utterance classification*, i.e., abstracting a specific message to the "activity" it refers to. In the rest of this section related works referring to each individual sub-problem will be discussed.

A. Process mining from conversational data

Generally, there is relatively limited research done in analyzing conversations using process mining techniques. Moreover, none of the techniques available in the literature is capable of a fully automated process mining analysis from both structured and unstructured conversational data.

Vakulenko et al. [2] used conformance checking [3] to evaluate a new model of information-seeking behaviour. Specifically, they represented their new behavioral model using a Petri net created from labelled and structured datasets and then used standard conformance checking techniques to observe the extent to which the model adhere to the dataset under examination. However, our work focuses on discovering the actual information-seeking behavioural model which is subsumed in conversational data by disentangling multi-thread conversations, labelling the utterances, and extracting topics. In general the focus of our work lies on the processing of the data and model discovery, rather than evaluating the behaviour of a preprocessed dataset.

Wang et al. [4] analyzed a dataset of structured conversations using conversation networks as well as process models. The goal of this was to determine, how different sub-graphs impact the effectiveness of the information-seeking threads. In their work, the data came from an Apple Support community, which originally included close to 50.000 discussion threads. However, the researches had to trim this dataset to 120 threads, because they manually had to perform utterance classification, which highlights the importance of a single framework capable of automatically performing this classification. In our work, instead, we present a fully automated framework capable of analyzing conversations coming from both structured (e.g., forums with conversations structured in threads) and unstructured (e.g., instant message systems) platforms.

The work presented by Di Ciccio et al. in [5] introduced a technique known as MailOfMine that could output a set of processes, given a collection of email messages. Their approach uses clustering to mine activities and tasks, similar to the approach taken in this paper as well. However, their approach also exploits features (such as Receiver and Attachments), that are specific to the structure of an email message. Our framework, instead, takes a generalised approach, without assuming any structured data, apart from message and author.

B. Message-stream disentanglement

Message-stream disentanglement, is the act of identifying separate conversations within a single stream of messages. This can often occur on online forums such as Twitch[3] or Gitter[4], where a single chatroom is not limited to one conversation at a time. Generally, methods for performing message-stream disentanglement can be divided into supervised and unsupervised approaches. The supervised approaches include works using siamese neural networks [6] to estimate the similarity between pairs of messages using message embeddings and contextually based features. Other works, like [7], exploit binary probabilistic classifiers to determine if a message is part of a previous conversation, or is itself the beginning of a new one. The features used to make these classifications, were based on the amount of time between messages, and cosine similarity between messages. An unsupervised approach include the work from Wang et al. [8] which assigned messages to conversations based on similarity functions, such as cosine similarity.

C. Utterance classification

Utterance classification has already been investigated extensively in literature, and its purpose is to reveal the intention of a given utterance based on different parameters. In the context

[3]See https://www.twitch.tv/
[4]See https://gitter.im/

of our work, utterance classification is the classification of a message in a conversation.

Bhatia et al. [9] proposed a classical machine learning approach classifying messages and exploiting features based on content, structural, user-information and sentiment value proposed to reveal that, utterances can be classified with an accuracy of up to 72.02%. They use 8 classes to categorize each message, in which 7 were first introduced in FIRE'10 [10]. The user-information based features employed in the work by Bhatia et al., are however strongly depending on the data set being informative with respect to the profile of the user.

Qu et al. [11] leveraged Bhatia et al. [9]'s taxonomy, but suggested 4 more classes, due to lack of categories. In their work, authors created a large scale annotated data set, MSDialog [11], extracted from a Microsoft Support forum. This data set is highly relevant to our work, and therefore both their taxonomy, seen in Table I as well as annotated data set will be used in our experiments.

Further work done by Qu et al. [12], revealed the importance of user intent prediction in the context of utterances, due to the advancement of CAs (Conversation Assistants). For CAs to be able to accurately identify user intent and make use of multi-turn conversations, Qu et al. suggested two approaches to identify a category of a given utterance. The first approach, similar to Bhatia et al. [9] extracts features based on structure, content and sentiment, and performs classical machine learning. The second approach uses a neural architecture to predict user intent. Moreover, these approaches offer a set of features which are better suited to the intention of our work – e.g. compared to Bhatia et al.'s approach – since they neglect features based on user profiles.

III. Mining process models from group chats

In this section we present the general framework for mining information-seeking conversations. This is decomposed into two research questions, that will be answered through our work.

A. Problem Statement

Currently there is limited research on the subject of applying process mining techniques and extracting process models from conversation data. Previous efforts to achieve this, used manually labelled data and analysed the process models extracted from such data. This, however, implies intense manual work which limits the data to be used by a large extend. Therefore our research proposes a general framework for extracting process models from information-seeking conversations, without the need of manual labelling. Therefore our research explores the following research questions:

- **RQ1**: Can we automatically extract process models from unlabelled information-seeking conversational data?
- **RQ2**: Can we derive meaningful insights from such process models?

B. Extracting process models from conversational data

In order to automatically analyze information-seeking conversational data with process mining techniques, it is necessary to pre-process the raw data and make it suited to the analysis. Specifically, event logs [1] are required to have, at least:

- The activity name that the event refers to. In our setting, the activity refers to the utterance of the message;
- The process instance (also called "case id") that an activity refers to. In our setting this refers to the discussion thread that a message belongs to;
- A timestamp, referring to the time when the action took place, i.e., when the message has been sent;
- Additional attributes which can be used to filter and refine the analysis. In our case, we will consider the set of topics that are discussed within in a thread.

The general idea for converting conversational data into process mining capable logs is graphically depicted in Fig. 2. The pre-processing comprises 3 steps (depicted as numbered arrows): *(1)* disentanglement of the messages in order to identify the thread of each message, *(2)* the identification of the utterance of each message, and *(3)* the discovery of topics referring to each discussion. Finally (depicted as step *(4)*), the event log can be processed using standard process mining techniques, for example to discover a process representation.

We can now discuss each component of the framework given in Figure 2. The framework is given a set of messages M. It then iterates over all messages and, for each message, it extracts the case id, the activity name and the set of topics of a message. With this information available, the framework composes an event which is then added to the log being generated which will eventually be returned and used in the analysis of the process model.

The framework consists of several components, which objectives are quite clear. Disentagle should assign to each message a case id, which can be one associated to a previous message or a new one. Utterance Classification is in charge of extracting the utterance, i.e., the activity, of a message. The set of activities that the function can extract is not constrained: different implementations of Utterance might map events to different numbers of activities. Finally, Topic Discovery has to identify relevant words in a message. Ideally, these words are the most contextually relevant, i.e., they reveal something about the content of the message and of the conversation. These topics are used to enrich the log, by giving the option to filter events based on specific topics. However, this step is not strictly necessary for the analysis with process mining, and could be left out if not needed.

IV. Conversational process mining on online information-exchange media

In this section we present a possible implementation of the components of the framework which are required in order to reveal useful information from conversational data.

The framework requirements for the input data are very limited: it is required that each message holds an attribute

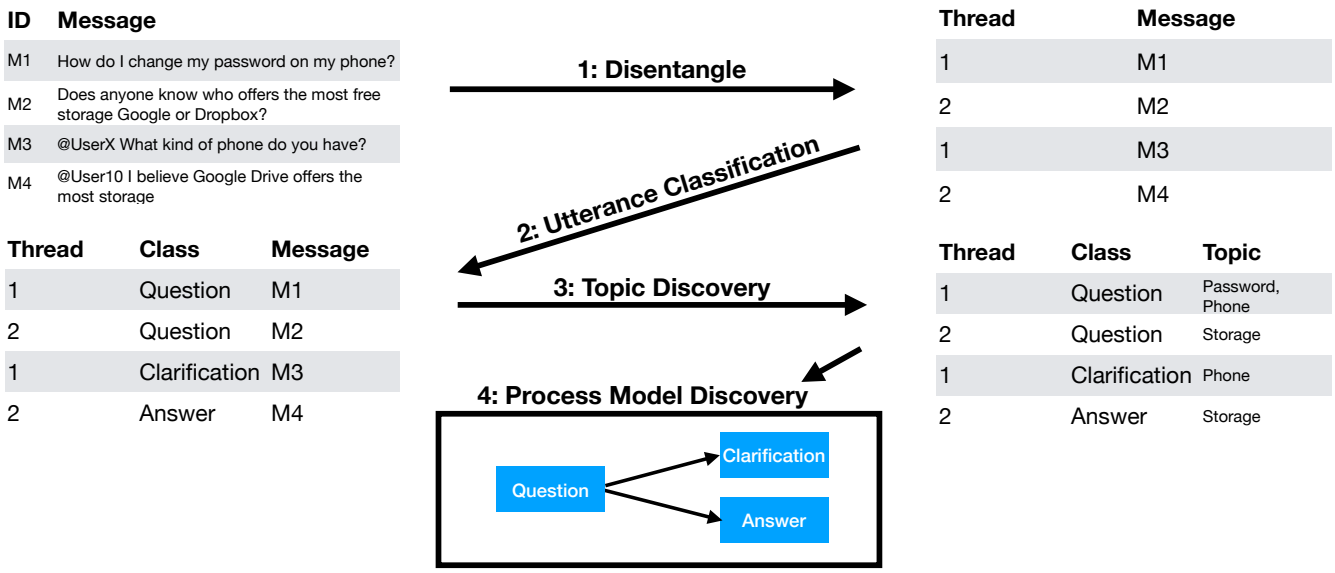


Fig. 2. General structure of the framework presented in this paper. All arrows, which are numbered, refer to the activities required to convert set of messages[6] into event logs to be analyzed with process mining techniques.

describing the *user ID*: typically a username referring to the author of the message itself, *date and time* of when the message has been sent, and the actual *content* of the post, expressed in natural language (e.g., English). With this information it is possible to disentangle and classify both the type of utterance and topics of the posts. The steps of the framework presented in this paper are depicted in Figure 2. In the picture, the arrows, which are numbered, refer to the manipulations of a given input in order to generate a corresponding output.

In the following subsections we describe a possible realization of each step of the framework. The technique has been implemented using Python and has been made publicly available along with the results.[5] [6]

A. Step 1: Message-Stream Disentanglement Using TF-IDF and Mentions

Due to the nature of information-exchange communities, it is fundamental to assume that not all data sets will be of perfect structure like the MSDialog data set. To obtain a dataset suitable for our process mining analysis it is required to disentangle the messages in order to group those belonging to the same conversation as reported in step 1 of Figure 2. As previously mentioned in Sec. II-B, there are multiple ways to disentangle message streams, however in our work, we decided to implement a new technique based on a combination of the TF-IDF technique [13] and the concept of *mentioning*. A mention refers to a user specifically targeting another user in a message. On many messaging platforms today, a mention is identified with an '@' sign (e.g., "@BillGates"). This makes it relatively easy to identify mentions by using regular expressions. The underlying assumption is that a mention is a strong indicator that a message belongs to a specific conversation, under the assumption that homogeneous set of users participate in the same conversation. Obviously, users could be contributors to multiple conversations at a time but in our context we decided to simplify the setting, assuming this not to be the case. Moreover, each conversation is assumed to have a max time-span i.e. from the first message of the conversation until the last there should be less than a certain number of minutes. This hyperparameter is subject to change depending on the contributors in the message stream. If a given message does not include a 'mention', it will be mapped to a conversation in a different way. In this paper we went with an unsupervised approach using TF-IDF and cosine similarity. Each message is pre-processed to remove punctuation, code blocks and upper case letters. Furthermore, we removed stopwords using NLTKs English stopwords[7], and stemmed words using NLTKs SnowballStemmer[8]. Then we mined the IDF from the set of pre-processed messages in the dataset, where each message is considered a bag of words. Once mined, we can represent each message as a vector of TF-IDF weights, and compute the distances between them in the vector space using cosine similarity. We used cosine similarity, due to its widespread use in NLP tasks. If the similarity between a new message and any of the current conversations is below a threshold, the message will be added as the initial message in a new conversation. This threshold should be altered depending on the dataset. For our experiments we set the threshold to 0.05.

B. Step 2: Utterance Classification

Given a dataset of structured conversation data, where each message belongs to a specific thread, we need to classify the utterance in order to elicit the "activity" subsumed by the user with a certain message. This activity is depicted in

[5]See https://github.com/lassestarklit/PM_for_Conversation
[6]See https://doi.org/10.5281/zenodo.3727599
[7]See https://www.nltk.org/book/ch02.html
[8]See https://www.nltk.org/howto/stem.html

TABLE I
TAXONOMY OF UTTERANCES USED IN THE PAPER, FROM [12].

Code	Label	Description
OQ	Original Question	The first question in the QA dialog.
RQ	Repeat Question	Another user repeats a previous Q
CQ	Clarifying Question	Request for clarification
FD	Further Details	Users or agents provide more details
FQ	Follow Up Question	Follow up questions about relevant issues
IR	Information Request	Agents ask for information of users.
PA	Potential Answer	A potential answer provided by agents.
PF	Positive Feedback	Positive feedback for working solutions.
NF	Negative Feedback	Negative feedback for useless solutions.
GG	Greetings/Gratitude	Users or agents greet each others.
JK	Junk	There is no useful information in the post
O	Others	Posts that cannot be categorized.

Figure 2 as step 2. As mentioned in Section II-C, this can be achieved in a variety of ways, exploiting different taxonomies. For this research, we tested AdaBoost [14], Naïve Bayes [15], Random Forest [16] as well as logistic regression [17] classifiers separately, given that these were shown to achieve good performance [12]. We first transform each message into a feature vector, based on content-based, structural and sentiment features, as outlined in [12]. These features include, but are not limited to, the utterance containing specific words or signs (indicating the utterance is a question), utterance similarity between one utterance and the initial utterance of a conversation and also the whole conversation (indicating relevancy of the utterance to the conversation), position of a message in a conversation i.e., index of the message in the list of messages which constitutes the conversation (indicating whether the utterance is a question/further details or an answer [12]), and sentiment score (indicating expressions of gratitude and identifying whether user feedback is positive).

Another parameter of this step is the taxonomy used for the classification, i.e., the set of labels that each classifier will associate a message to. This taxonomy will have a high impact on the final process models, as these labels will define the activities of the processes. With this in mind, we used the taxonomy seen in Table I, which originally comes from [12].

C. Step 3: Conversations Represented by Topics

Step 3 of the data processing in the framework, as depicted in Fig. 2, is to extract the topics discussed in a message. In the context of this paper, a topic is a set of labels that somehow captures the content of the individual message. In the examples from Figure 2 (messages in the top-left table) topics could be labels like "Storage", "Password" and "Phone". By extracting these topics for each thread, we can enrich the process mining capable dataset with information regarding, not only the structure, but also the content of the discussion.

In our implementation we mined the Inverse Document Frequency (IDF) *a priori*, and computed TF-IDF at runtime. The words with the highest TF-IDF scores from the first message in the thread was then used as the topic for the whole thread. We chose to limit this to three topics per thread i.e. the three highest scoring words were used as the topics. The TF-IDF was computed on the pre-processed utterances. Similarly to Step 1, the stopwords, code and punctuation was removed, and the words were stemmed using NLTK for python.

D. Step 4: Process Mining Analysis

For the process mining analysis, we used the tool Disco[9]. The tool, which supports control-flow discovery and trace filtering, is designed to be interactively used by exploring the process maps at different levels of details (i.e., showing all behavior vs showing just the most important one). For our experiments we will test different parameters configurations and report the corresponding maps.

V. EXPERIMENTS & RESULTS

In this section we present the results of running three datasets through our proposed framework, to evaluate our implementation and the process models gained from the datasets. One synthetic dataset consisting of 7 conversations, wa constructed, as an example of what the framework accomplishes. The dataset is made publicly available[10]. The other two datasets used in the experiments are MSDialog [11] and the Gitter Free Code Camp dataset[11]. All experiments were run on a standard machine running on MacOS Catalina with 8 GB RAM, a 3,1 GHz Intel Core i5 CPU and Python 3.7.6.

A. Syntehtic Dataset

The first experiment we present was conducted to present the reader to a concrete example, of how our framework maps conversations to process models. We first generated a synthetic dataset by creating different patterns of conversations (i.e., how an information-seeking message could introduce different types of conversations, such as $OQ \rightarrow PA \rightarrow GG \rightarrow GG$), and hereafter improvising the textual content. The dataset consists of 7 conversations exposing different patterns.

As emphasized previously; structure, relevance and formality can vary substantially depending on source/platform of which the conversation takes place. Thus, in this experiment, we trained 3 models on 5 of the 7 conversations to evaluate the first 2 steps of the framework. For testing purposes, we interleaved the remaining 2 conversations. Table V-A presents the content of the messages in the first column. They are represented in an order computed by the disentanglement step (cf., column *Disen. pos*), which also happens to be the correct one. The original/scrambled order is shown in the column *Abs. pos*. We evaluated the different classifiers and concluded that the classifier suited best for this dataset was Adaboost. Looking at true and predicted classes, the only misclassification is a message of further details (i.e., 'FD'), which is classified as potential answer (i.e., 'PA').

From this small synthetic dataset we report an accuracy message classification of 88.8% and 100% accuracy on thread disentanglement, however *mentionings* appeared to be a vital factor to the disentanglement.

[9]See https://www.fluxicon.com/disco
[10]See https://doi.org/10.5281/zenodo.3727599
[11]https://www.kaggle.com/freecodecamp/all-posts-public-main-chatroom

TABLE II
TRUE AND PREDICTED CLASS LABELS AND THREAD LABELS FOR A SMALL SYNTHETIC DATASET

Content	True class	Pred. class	Abs. pos	Disen. pos
Does anyone know who offers the most free storage Google or Dropbox?	OQ	OQ	1	1
@User10 I believe Google Drive offers the most	PA	PA	2	2
cool thanks @User11 :)	GG	GG	5	3
You're welcome @User10	GG	GG	7	4
How do I change my password on my phone?	OQ	OQ	3	5
@UserX What kind of phone do you have?	IR	IR	4	6
@Agent I have an iPhone 3	FD	*PA*	6	7
@UserX you can follow the instructions from this link	PA	PA	8	8
It seems like it works @Agent :) Thanks!	PF/GG	GG	9	9

The last part of the framework consist of a process model analysis. The model related to the synthetic dataset, is seen in Fig. 3. The process map reported in the figure contains one node for each activity (plus circled nodes indicating the beginning and the end of the process). Activities are connected with edges which indicate that the two activities are observed one immediately after the other. The thickness of the edges and the color-scale of the activities indicate the frequencies (i.e., thicker edges refer to direct successions being observed more often, darker activities refer to activities happening more frequently). Additionally, both activities and edges have numeric values indicating absolute frequencies and median time, however for this map, since it is synthetic, these times do not reveal any useful information.

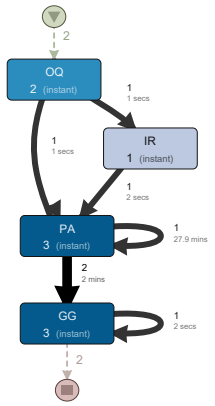


Fig. 3. The process model from the synthetic dataset

B. Complete Framework Test on Gitter Dataset

In our work, we tested the process of the entire framework on an unstructured dataset extracted from 3 months of conversations on the Gitter Free Code Camp[12], in order to answer RQ1 (cf., Sec. III-A). The dataset is freely available as a `.csv` file, and is structured as a long conversation, referred to as a "group-chat". Each message comes with appertaining attributes such as *user id* and *timestamp*. Applying the first component of the framework, the *Message-Stream Disentanglement* described in Sec. IV-A, to the dataset will provide a structure, such that each message is either rejected or assigned as an event in a trace in the event log. Furthermore, we apply the classification component, referred to in Sec. IV-B. This process associates each event to an activity, which will be used in the rest of the analysis and for this activity we used Logistic Regression. The last step is the *Message-Topic Discovery*, described in Sec. IV-C, to populate each trace with words of significance in relation to the respective stream of messages. After processing the dataset, the set of traces represent conversation process models, which can be analyzed to discover meaningful insights from the

[12]https://www.kaggle.com/freecodecamp/all-posts-public-main-chatroom

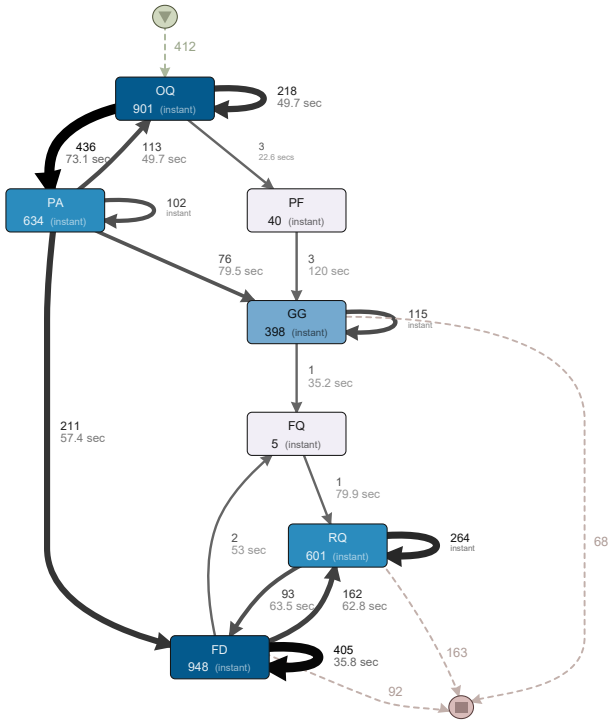
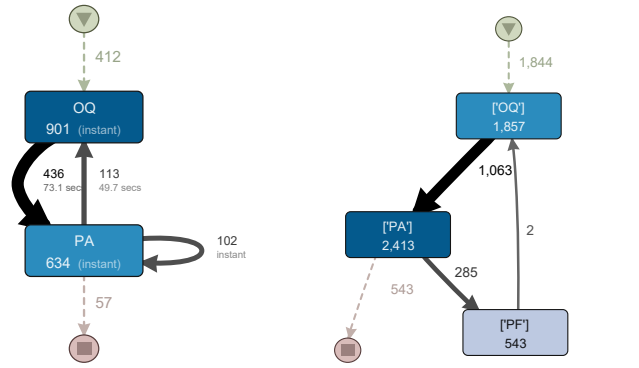


Fig. 4. The process model of event log from the Gitter dataset

online message-exchange platform. In total, the event log contains 20.485 events, each referring to 1 if 7 activities distributed over 11.491 traces.

Our analysis of the extracted process model reveals some important insights regarding the process of requesting information. In the rest of the analysis we filtered only traces referring to information-seeking conversations: the first activity has to be an actual question (i.e., 'OQ' as in Tbl. I), thus ending up with 3495 traces (about 30% of all cases). Looking at the efficiency to which one can expect a solution to the proposed question, the data reveals that in 10% of the cases the initial question is solved in the immediately following utterance. Figure 4 reveals the process when an information-seeking utterance initializes the conversation.

From the median time of the mined process, emphasized in the secondary value in Fig. 4, we see that the time between the beginning of an information-seeking utterance to a potential answer (i.e., 'PA') is 73 sec. The most frequent flows in Figure 4 follow scenarios of information-seeking-utterance to solution patterns, which was expected before analyzing the process model. The pattern, which appears as the most common in the dataset is $OQ \rightarrow PA \rightarrow FD \rightarrow RQ$. The flow is initialized with a question, and directly followed by a potential answer. However further detail is provided (the self loop in FD indicates that multiple informational utterances are exchanged). The repeated question ('RQ', a question similar to the original question, asked by another user) that ends the flow indicates that the original question is not resolved. Another common pattern is $OQ \rightarrow PA \rightarrow GG$ and it indicates a successful flow seen from an information-seeking perspective, as an original question is followed by a potential answer,



(a) Most common process in Gitter (b) Most common process in MSDialog

Fig. 5. Highly filtered and aggregated Disco transition graphs for both Gitter and MSDialog datasets.

which must be assumed to be correct, since the following utterance is a gratitude (i.e., 'GG').

Investigating the most abstract version of the process map (i.e., setting 0% for both parameters in Disco), we observe that the most common flow consists of an original question directly followed by a potential answer, as seen in Fig. 5(a). This observation supports the idea that seeking information in Gitter, does indeed have a beneficial effect.

C. Deriving Insights from MSDialog Dataset

In this section we report the results of the investigation on a subset of the MSDialog dataset, in order to answer **RQ2** (cf., Sec. III-A). This dataset contains labels for the utterances, as well as structured conversations and topics. We can therefore rewrite the format of this data, to present process models from MSDialog, that are free of classification errors. In other words, this experiment focused mostly on the analysis of the results.

The process model extracted from the data can be seen in Fig. 6. For this analysis we discarded conversations lasting longer than one year (these are often terminated conversations, where a new user posts something several years later). Furthermore, the graph was constructed in Disco with high levels of filtering, to be able to construct a comprehensible process map. We also considered only discussions starting with an original question (i.e., 'OQ') Therefore, the event log used for the analysis contains 8455 events, referring to 251 activities distributed over 1858 traces.

The process maps reveals that roughly half of the conversations in the dataset, starts with an original question (i.e., 'OQ') and receives a potential answer (i.e., 'PA') within 35 hours on average. The other half of conversations contains an original question, that is followed up by some kind of information request (i.e., 'IR') or some further details (i.e., 'FD'). More generally, the data reveals a median case duration of 3.2 days, with an average of 4.6 messages per conversation.

Lastly, in Fig. 5(b) we filter the graph, to reveal the most common process in the dataset (i.e., setting 0% for both parameters in Disco). We can see that most of the time a question from a user receives a useful answer straight away, and often a user responds with positive feedback.

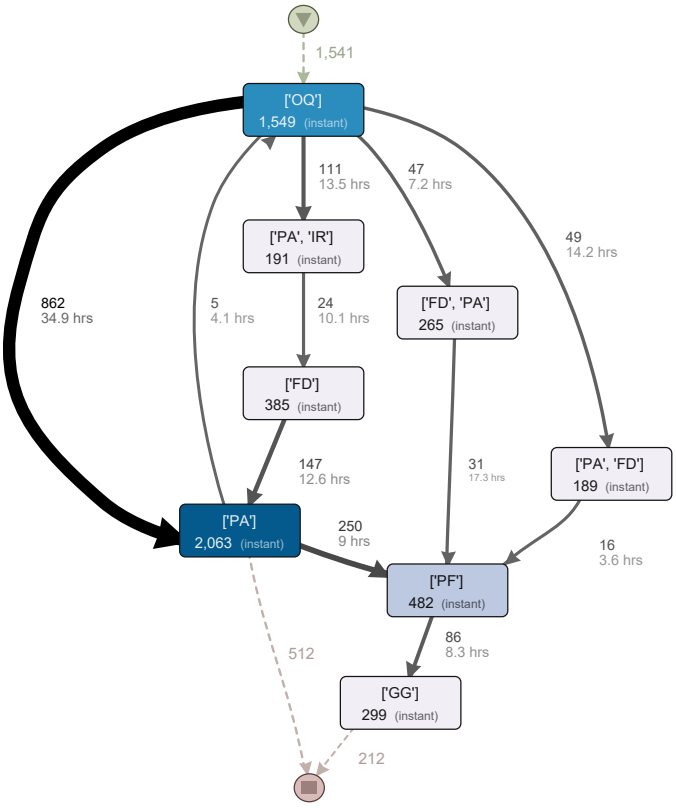


Fig. 6. The process model of the annotated MSDialog dataset

TABLE III
CONVERSATION METRICS FOR VARIOUS TOPICS IN MSDIALOG

Topic	Median Conversation Duration	Msg. Per Conversation
Windows 7 and 8	4.9 days	4.49
Skype	3.3 days	4.41
Windows 10	2.5 days	4.82
Office	2.0 days	4.69

By comparing the two most abstract process maps as in Fig. 5, referring to the two datasets examined in the paper it is interesting to see that the behavior of the users on the two platforms is quite similar. Still, it is relevant to note that in MSDialog potential answers are sometimes followed by positive feedback. This could be due to difference of formality in a instant-messaging platform and a forum or the simple reason that an equal amount of messages are followed up by a negative/positive feedback in the Gitter dataset.

D. Analysis of Conversation Topics

Lastly, we present conversation metrics on conversations referring to specific topics. From the MSDialog dataset, we utilize their annotation and the user-specified categories herein. We constructed 4 aggregated classes: Windows 7 and 8; Skype; Windows 10; and (Microsoft) Office. The metrics used for evaluation are median conversation duration and messages per conversation. The results are presented in Table III, which is sorted in descending order of conversation duration. From this, it is interesting to note, that questions regarding Windows 10 are resolved twice as fast as questions on Windows 7 and 8.

TABLE IV
CONVERSATION METRICS FOR VARIOUS TOPICS IN GITTER DATASET

Topic	Median Conversation Dur.	Msg. Per Conversation
IDEs	9.9 minutes	7.75
Programming Languages	9.8 minutes	7.68
Microsoft Products	9.8 minutes	6.00

Similarly, we constructed three aggregated classes for the Gitter dataset: Programming languages (HTML, CSS, JavaScript, JQuery, Nodejs, Rails); IDEs (Sublime Text, Vim, Emacs, Visual Studio); and Microsoft Products (Windows 7 and 10, Microsoft, Skype, Office). As reported in Table IV the conversation durations are quite similar for all topics. Additionally, the average number of messages per conversation is slightly different for the different topics.

VI. DISCUSSION

Going back to the original research questions stated in Section III, we found **RQ1** to hold true. We were able to show through experiments in Sec. V-A and V-B how we could extract process models from information-seeking discussions datasets. Specifically, in the experiment in V-B we applied the framework to an entirely unlabelled, unstructured, and real dataset, and were able to produce process models from this. To our knowledge, extracting process models from unlabelled and unstructured information-seeking conversation dataset is something that has never been reported before. Furthermore, we also found **RQ2** to hold true from experiments in Sec. V-B, V-C and V-D. These experiments showed how it was possible to extract information regarding the process of answering questions in two different settings. Additionally, different metrics of these processes such as conversation duration, were analyzed based on the topics covered in the discussions.

Our work has revealed that the idea of converting conversational data into process mining reports is an important factor in understanding and learning the structure of a human-to-human conversation. The behavior of a conversation is highly connected to the platform used for exchanging the messages. Specifically, with our framework we are able to reveal structure out of an unstructured data. Moreover, we have experienced how different types of online message-exchanging platforms can lead to different process flows, and how topics on the same platform might unfold in terms of duration. These observations amongst a number of others which this framework is capable of revealing, will help to understand and exploit parameters, such as the efficiency of a platform.

While this framework proved to be capable of answering our research questions, and reveals benefits, we were still able to identify certain limitations. Firstly, the framework is sensitive to the choice of parameters like choice of disentanglement function, utterance classification function and topic extraction function. Furthermore, our research was applied to datasets of information-seeking conversations (i.e. there is a request and exchange of meaningful information between users with a goal to achieve). Therefore, the application of our proposed framework in more informal inconsequential conversations might be limited.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented a general framework for extracting process models from unstructured information-seeking conversations. To accomplish this, we divided our framework into the three steps: disentanglement, utterance classification and topic extraction, and we presented an algorithm for this. Furthermore, we provided an instantiation of our framework which uses cosine similarity for disentanglement, a classifier selected among a pool for utterance classification and TF-IDF for topic extraction. This instantiation was tested on a large unlabelled and unstructured chat dataset. Results showed that both the research questions identified hold true. Additionally, however, limitations were pointed out concerning the sensitiveness of the approach to parameters and the application domain (which has been limited to information-seeking conversations).

Future work will make the disentanglement and the utterance classification steps more robust, for example adopting entropy-based measures or ensemble classifiers. Another future work direction includes testing the framework on different datasets where the conversations are not so much goal-oriented and where chatbots are involved.

REFERENCES

[1] W. van der Aalst, *Process Mining*. Springer, 2016.

[2] S. Vakulenko, K. Revoredo, C. Di Ciccio, and M. de Rijke, "QRFA: A Data-Driven Model of Information-Seeking Dialogues," in *European Conference on Information Retrieval*. Springer, 2019, pp. 541–557.

[3] J. Carmona, B. F. v. Dongen, A. Solti, and M. Weidlich, *Conformance Checking*. Springer, 2018.

[4] G. A. Wang, H. J. Wang, J. Li, A. S. Abrahams, and W. Fan, "An analytical framework for understanding knowledge-sharing processes in online Q&A communities," *ACM TMIS*, vol. 5, no. 4, p. 18, 2015.

[5] C. Di Ciccio, M. Mecella, M. Scannapieco, D. Zardetto, and T. Catarci, "Mailofmine–analyzing mail messages for mining artful collaborative processes," in *International Symposium on Data-Driven Process Discovery and Analysis*. Springer, 2011, pp. 55–81.

[6] J.-Y. Jiang, F. Chen, Y.-Y. Chen, and W. Wang, "Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking," in *Proc. of NAACL-HLT*, 2018, pp. 1812–1822.

[7] E. Mayfield, D. Adamson, and C. P. Rosé, "Hierarchical conversation structure prediction in multi-party chat," in *Proc. SIGDIAL*, 2012.

[8] L. Wang and D. W. Oard, "Context-based message expansion for disentanglement of interleaved text conversations," in *Proc. of NAACL-HLT*. Association for Computational Linguistics, 2009, pp. 200–208.

[9] S. Bhatia, P. Biyani, and P. Mitra, "Classifying user messages for managing web forum data," in *Proc. of WebDB*, 2012.

[10] C. Hori and T. Hori, "End-to-end conversation modeling track in dstc6," *arXiv preprint arXiv:1706.07440*, 2017.

[11] C. Qu, L. Yang, W. B. Croft, J. R. Trippas, Y. Zhang, and M. Qiu, "Analyzing and characterizing user intent in information-seeking conversations," in *Proc. of SIGIR*, 2018, pp. 989–992.

[12] C. Qu, L. Yang, W. B. Croft, Y. Zhang, J. R. Trippas, and M. Qiu, "User intent prediction in information-seeking conversations," in *Proc. of ACM SIGIR CHIIR*, 2019, pp. 25–33.

[13] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[14] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[15] D. J. Hand and K. Yu, "Idiot's bayes—not so stupid after all?" *International statistical review*, vol. 69, no. 3, pp. 385–398, 2001.

[16] T. K. Ho, "Random decision forests," in *Proc. of IEEE ICDAR*, vol. 1, 1995, pp. 278–282.

[17] D. G. Kleinbaum, *Logistic Regression*. Springer, 1994.