May 2023

NESTED MULTI-LEVEL CONTEXT RESOLVER

Marcelo Yannuzzi

Chiara Troiani

Jean Diaconu

Hervé Muyal

NESTED MULTI-LEVEL CONTEXT RESOLVER

AUTHORS:
Marcelo Yannuzzi
Chiara Troiani
Jean Diaconu
Hervé Muyal

ABSTRACT

Team members working on a project need to constantly pull, read, process, merge, and create information across different tools and applications when collaborating on the project. Team members face complex and time-consuming challenges to stay up to date and focus on the context that matters to them. Techniques described herein provide for automated extraction of the context that matters to each worker involved in an upcoming event (e.g., prior to a customer meeting, a sprint review, a project milestone review, etc.) by using a nested multi-level context extraction and resolution technique.

DETAILED DESCRIPTION

The number of virtual meetings, chat spaces, emails, shared documents, issue and project tracking software boards, confluence pages, and other sources of information typically tend to increase during the lifecycle of a project. Therefore, project members ranging from implementors to decision makers need to constantly pull, read, process, merge, and create information across different tools to collaborate. Although various techniques have attempted to provide a more cohesive integration among different collaboration tools, team members still face complex and time-consuming challenges to stay up to date and focus on the context that matters to them.

In addition, some team members might be involved in two or more projects simultaneously. Virtual meetings that require their participation often overlap across projects or need to be scheduled at late hours, as teams may need to collaborate across different time zones. Although the topics that are going to be addressed in a virtual meeting as well as the role and expectations for a given participant might be inferable by correlating contextual information available in the different chat spaces, emails, previous meeting recordings, the list of attendees, the title of the meeting, and/or other sources, quite often various attendees simply lack such context. For instance, an attendee may not have sufficient time to process all the notifications, data, and documentation sent over the various channels, since the attendee might need to multi-task across various work threads or projects. In turn, many virtual meetings simply lack a well-defined agenda, so skipping them might be a risk, since important topics might be discussed, participation of the invitee might be expected, or decisions might be made without the invitee being present.

Overall, the challenge is trying to be "everywhere" by switching from one tool to another, and trying to be on top of every email, virtual meeting, or message posted in a space. The underlying problem is the inability to extract, correlate, infer, and present context digests in a uniform manner across the various tools that teams usually use to collaborate.

To build such context (e.g., prior to a customer meeting, a sprint review, or a project milestone review), a person needs to process data from each input source individually, make decisions about the thematic and temporal relevance of such data, identify and correlate specific entries across the different sources, extract the context that matters (tailored to each team member's role), and prioritize it. Each team member may need to repeat this same pattern "manually" and "individually" over and over again repeatedly.

Techniques described herein enable automated detection and dynamic aggregation of entries that are linked thematically, and temporally, across the various information systems and tools that a team typically uses to collaborate, including, for example, chat spaces, meeting systems, emails, repositories with shared documentation, issue and project tracking software boards, or confluence pages. Such entries may be discovered, analyzed, correlated, prioritized, and presented as context digests in a uniform manner, and they may be personalized to each worker using a nested multi-level context resolution technique.

The techniques described herein are targeted towards automatically detecting upcoming events, such as virtual meetings, standups, or customer meetings, which might be scheduled in one or more tools (e.g., in calendars, in meeting systems, etc.) and contextually identifying, analyzing, selecting, and filtering previous notifications, messages, and/or documentation that might be relevant, from the various sources of information that a worker typically uses. The focus herein is mainly on the continuous discovery, amalgamation, and extraction of the context that matters to each worker (e.g., an attendee involved in an upcoming meeting). Techniques described herein may not only help workers to gain focus and reduce the notification fatigue that many workers currently suffer, but also serve as input to external virtual assistants and/or recommendation systems.

Nested Multi-Level Context Resolution Process

Figure 1, below, illustrates an example system in which the techniques described herein may be implemented.

As illustrated in Figure 1, in a first step (1), a set of collaboration systems and tools may be used as inputs to a Resolver 101. Such inputs may include information contained in chat spaces 102, in a meeting management tool 103, in emails 104, in documentation spaces 105, in file sharing systems 106, in developer tools 107, or other sources of information. A set of connectors 108 might be developed to ingest relevant information from each source. Such connectors may comprise application programming interfaces (APIs), bots, scripts, or other elements enabling Resolver 101 to pull or subscribe to relevant notifications (e.g., a new email arrived, or there is a new message in a relevant space, or a file was updated, or a new space was created), pull or subscribe to upcoming events (e.g., a new meeting is scheduled, or a new milestone is set, or a new deadline is defined), and selectively parse and normalize the corresponding contents across the various data sources.

Connectors 108 may be configurable, and may provide analysis and persistence features (e.g., enabling threading and sometimes keeping state while processing data for specific themes, project, or workers (e.g., employees, subcontractors, etc.)). For instance, this may help to compute the "deltas" between two consecutive reads. In some cases, a subscription to a notification process might not exist, so a connector may need to periodically pull information from a source, check for updates, and extract the "delta" in the data retrieved.

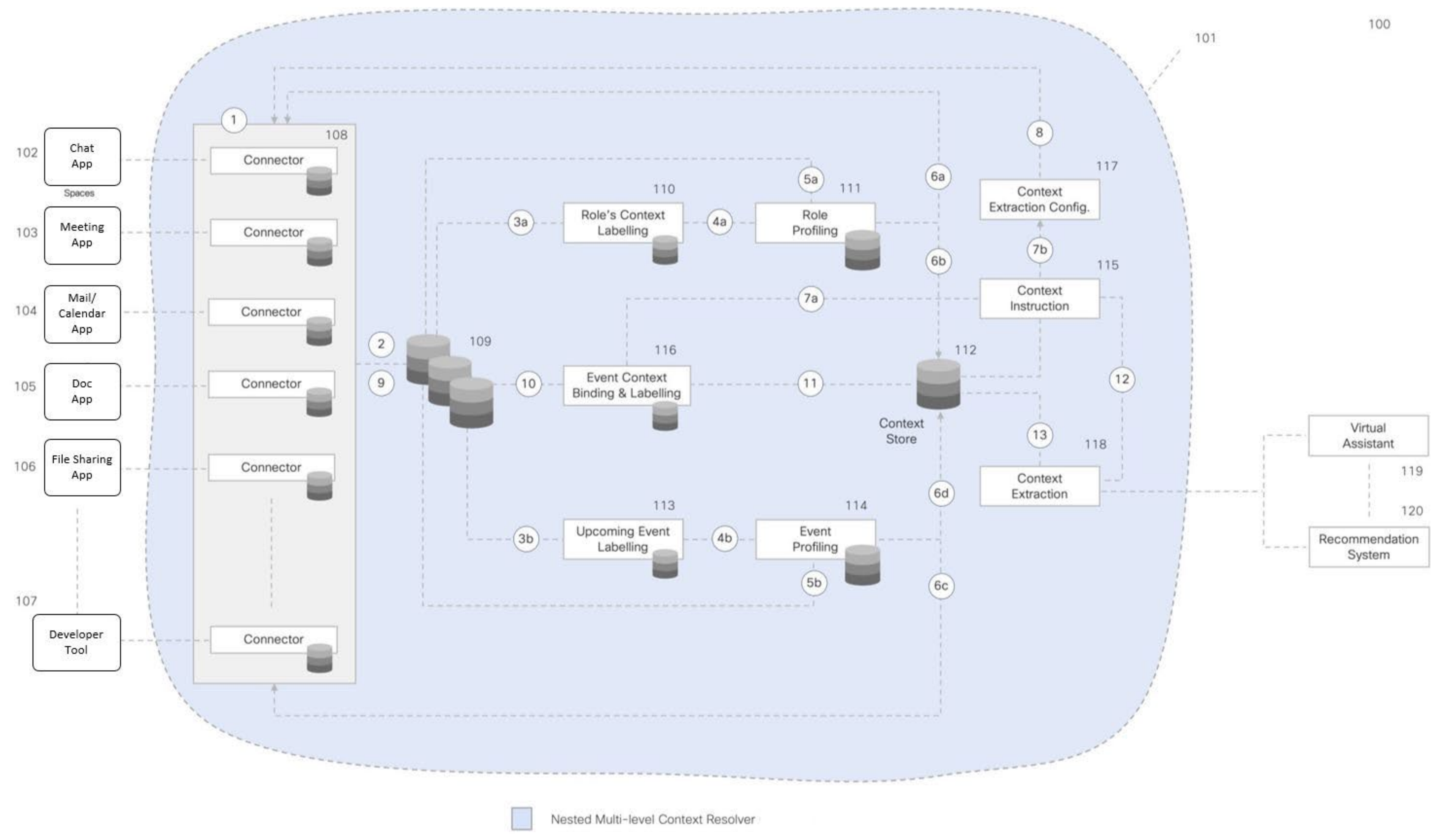

Figure 1: Example System for Performing a Context Resolution Process

Continuing from step (1), in a second step (2), the data gathered concurrently by collectors 108 may be sent to a data lake 109. In steps (3)-(6), Role's Context Labelling module 110 might specifically analyze and label signals (data) from data lake 109 for the different workers involved in the system at step (3a). The outcome of such labelling might be made available to Role Profiling module 111 at step (4a), and Role Profiling module 111 may identify and profile the role of the workers in the different projects or work streams at step (5a). For instance, the profiling process may detect various possible identifiers linked to the same worker (e.g., the worker's ID, email, git user ID, a nickname, a voice in a recording even if the worker joined a virtual meeting in a room without login into the meeting, etc.).

Hence, the combination of steps (3a)-(4a)-(5a) may enable Resolver 101 to create the first level of context about each specific worker. More specifically, such profiling may combine information from various sources to detect whether a worker is with sales, engineering, marketing, or another department. It may also help with understanding the role of a worker in a given project or work track.

Additional sources of information complementing the example inputs 102 to 107 might be used as well, enabling further profiling features. For instance, identifying the worker's location may denote behavioral patterns, including cultural and responsiveness features, the worker's seniority, role within the company, years with the company, role in the project, responsibilities, etc. Examples of sources of information that may be used to this end may include human resource (HR) tools, tables in spreadsheet files, a team table with roles in team workspaces, a board in issue and project tracking software, etc.

Other more insightful features might be inferred and scored from the tone and context extracted by live data through connectors 108, such as the worker is "very vocal," is a "leader," has "execution focus," has "sales focus," is a "decision maker," or is "strongly technical." All these features may help to profile and automatically infer each worker's role and focus areas within a project. More importantly, all these inferences and profiling may help building the second, and third level of contexts, as described below.

Terms and conditions in the use of Resolver 101 may prevent using it for other purposes than those consented to by the users (e.g., avoiding tracking and controlling the workforce activity). Role's Context Labelling 110 may also keep state and gather and combine signals from the same source or other sources. For instance, Role's Context Labelling 110 may detect the presence of a new team member in a project, identify the worker's ID, and request Role Profiling module 111 to instruct connectors 108 to collect data for that specific worker ID in step (6a). In turn, Role's Context Labelling 110 may now be prepared to combine and label signals received for the worker. For instance, the worker may have been added to a team table in a team workspace, with a specific role in the project, may now have assignments in an issue and project tracking software, may have been added to various chat spaces, and invited to weekly stand-up meetings. All these different signals may be consistently labelled, which may help Role Profiling module 111 determine that the new incorporation to the team has a "developer" role. Role Profiling module 111 may push the outcome of such profiling to Context Store 112 in step (6b).

According to techniques described herein, Role Profiling module 111 may also be able to determine the role with fine granularity, and even disambiguate between the worker's profile at a company level, which, for example, might be "full-stack developer," to the precise function in the "context" of a specific project, which might turn out to be "Site Reliability Engineer lead" or "DevOps lead." The capacity to disambiguate and correctly profile the worker's role within a specific project may be possible thanks to the collection and labelling of signals performed by elements 108, 109, and 110, the analysis and profiling carried out by 111, and persisted in Context Store 112. This profiling process is continuous and the outcomes may vary over time and per project or work track, for each worker.

In parallel to processes (3a)-(4a)-(5a)-(6a)-(6b), an Upcoming Event Labelling module 113 might also analyze and label signals (data) from data lake 109 in step (3b). These signals may specifically focus on detecting upcoming events (e.g., virtual meetings, workshops, customer meetings, product releases, demos at conferences, etc.). The outcome of such labelling might be made available to Event Profiling module 114 (4b), which may now proceed to identify and profile the upcoming event in step (5b). Hence, the combination of steps (3b)-(4b)-(5b)-(6c)-(6d) may enable Resolver 101 to create a second level of context, in this case about upcoming events that a worker might be involved in.

Like in the case of the processes used by Resolver 101 to build the first level of context, the overall combination of steps (3)-(4)-(5)-(6) may enable Resolver 101 to maintain two levels of relevant context stored in Context Store 112.

Nested Context Resolution

In steps (7) to (11), except for admission control and arbitration techniques that might be required to avoid inconsistent configuration requests to connectors 108 in steps (6a) and (6c), the first two levels of context resolution described above may run as independent threads. The third level of context extraction and resolution described in steps (7) to (13) may run as a nested process within the previous two levels.

More specifically, a Context Instruction module 115 may continuously analyze the information in Context Store 112. When a new (upcoming) event is detected, Context Instruction 115 may require instructing Event Context Binding & Labelling module 116 to initiate a labelling process for that specific event bond to the worker’s IDs (e.g., the participants) detected for the event in step (7a). Hence, Event Context Binding & Labelling 116 may not only use the context and labels produced during the first and second levels of context creation in Context Store 112 but may also push new labelled data and the nested context created to Context Store 112.

In some cases, the context observed by Context Instruction 115 through Context Store 112 might be ambiguous. For instance, message threads captured within the same chat space might be intertwined (e.g., two different threads of conversation might get mixed in the same chat). In such cases, Context Instruction 115 may request Context Extraction Configurator 117 to fine tune the temporal resolution, add filtering, or disambiguate contexts across ambiguous signals in step (7b). Context Extraction Configurator 117 may now instruct connectors 108 to adapt the temporal resolution of the data collected and/or look for specific elements enabling such disambiguation in step (8). For instance, this may allow for going back to messages previously posted in a chat space, processing the audio and/or the transcript of a previous meeting, or searching for specific content in a shared file, etc. This capability may allow for adjusting the temporal resolution of the context inference techniques described herein and/or adding additional signals (context) that might statistically help during a disambiguation process. Multi-threading techniques and the capacity to keep state may enable such features in connectors 108.

As new signals are created and collected by connectors 108, the signals may be pushed to data lake 109 in step (9). Event Context Binding & Labelling 116 may now proceed in step (10) to bind and label relevant signals involving various possible identifiers denoting the same upcoming event (e.g., a meeting ID, a key and repeated date, an acronym such as CLUS or EBC, etc.) obtainable through the second level of context created and various possible identifiers denoting the same worker (e.g., as detailed before, the worker's ID, email, git user ID, a nickname, a voice in a recording even if the worker joined a virtual meeting in a room without login into the meeting, etc.) obtainable through the first level of context created.

As described above, the result of the binding and labelling process may now be made available to Context Store 112 at step (11). Moreover, the specific labels used may also be available to Context Instruction 115. In steps (12) and (13), Context Extraction module 118 may now identify the new labels created by Event Context Binding & Labelling process 116 (e.g., through Context Instruction 115 in step (12)). It may also have access to Context Store 112, and thus, it may proceed to create the third level of context targeted in this invention, that is, one that extracts and derives specific context prior to an upcoming event tailored to each worker at step (13).

As described above, step (12) may be bidirectional, so Context Extraction 118 may leverage Context Instruction 115 to disambiguate blurred context after the binding and labelling process performed in step (11). Moreover, the context inferred by Context Extraction 118 may serve as input to external systems to Resolver 101, such as a Virtual Assistant 119 or a Recommendation System (120).

Figure 2, illustrated below, depicts an example showing one possible use of Nested Context Resolver 101.

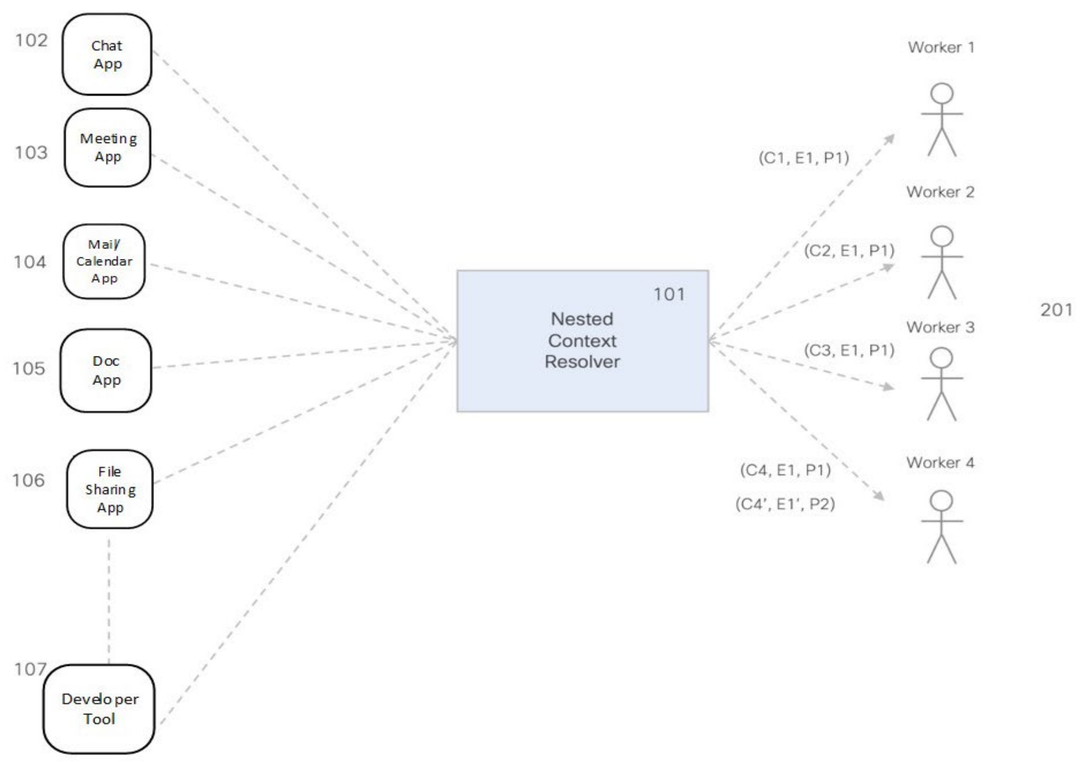

Figure 2: Example Showing Use of Nested Context Resolver

In this case, a worker 1 may receive a context digest C1, for event E1, as part of project P1. The automated identification and profiling of worker 1 and the worker's participation in project P1 may be obtained by the first level of context extraction described above in Figure 1. Similarly, the detection and profiling of event E1 may be obtained by the second level of context extraction described above, while the overall binding of the triplet (C1, E1, P1) to worker 1, as well as the contents carried in the digest C1, may be obtained by the nested context extraction mechanisms above described. Similarly, workers 2 and 3 may receive the triplets (C2, E1, P1) and (C3, E1, P1), respectively, for the same event (E1) and project (P1). However, worker 4 may simultaneously participate in two projects, P1 and P2, and therefore, may receive two independent digests, C4 and C4' tailored for worker 4, for an upcoming event E1 as part of project P1, and another upcoming event E1', as part of project P2.

The example embodiments described above may leverage natural language processing (NLP) techniques to collect textual signals from various sources, such as names and components in project tracking software, email subjects, email bodies, the name assigned to chat spaces, messages within those space, meeting titles, list of attendees, etc.

They may also leverage video recordings, and techniques like optical character recognition (OCR) applied to video recordings or demos (e.g., in presentations that might not be available through the different input sources, such as 102 – 107, which might be the case when slides and/or the demos are owned and presented by an external organization). They may also leverage audio, audio recordings and analysis techniques applied to the audio.

In one embodiment, Resolver 101 may even generate context digests and alerts to a worker, even when the worker might not have any of the tools 102 – 107 opened or in use. In another embodiment, Resolver 101 may differentiate between recurrent events (e.g., a weekly review) and non-recurrent events, and it may adapt the context digests accordingly. It may also detect whether a worker is on paid time off or other types of leave, and it may automatically stop the creation of contexts during such periods.

Moreover, context digests may be created and dynamically updated for any upcoming event, and such context may be available for historian analysis and post-processing after the event occurred. Various training techniques may be used to increase the accuracy of the two levels of profiling described above as well as the nested binding and context resolution implemented by Resolver 101.

According to one technique, the context digests may be presented daily, weekly or in a configurable manner. For instance, they may be customized per project, or based on variable frequency (e.g., increase frequency as the date of the event approaches).

In another embodiment, some of the capabilities at the connector-level, as well as at the three levels of context generation described above, may leverage large language models, and may use Generative Pre-trained Transformer (GPT) architectures (e.g., both reading and writing purposes). To this end, the ingested data may be indexed to allow precise queries from Resolver 101. Additionally, these language models may be fine-tuned with a human in the loop (e.g., leveraging Reinforcement Learning from Human Feedback RLHF techniques) to improve performances, minimize bias and align the models to each of the 3 Context resolution tasks.

In summary, techniques described herein provide for automated extraction of the context that matters to each worker involved in an upcoming event (e.g., prior to a customer meeting, a sprint review, a project milestone review, etc.) by using a nested multi-level context extraction and resolution technique.