Dialogue Management as Graph Transformations

Nicholas Walker, Torbjørn Dahl and Pierre Lison

Abstract We present ongoing work on a new dialogue management framework using *graphs* as core representation for the current dialogue state. Dialogue management tasks such as state tracking and action selection are framed as sequences of *graph transformations* that repeatedly update this graph based on incoming observations. Those graph transformations are expressed using a graph query language, making it possible to specify all dialogue management operations through a unified, declarative syntax. We argue that graphs are particularly well-suited to model the dialogue state of complex, open-ended domains. In contrast to traditional dialogue state representations that are limited to fixed, predefined slots, graphs can naturally express dialogue domains with rich relational structures and variable numbers of entities to track. We describe how dialogue state tracking and action selection can be modelled in such graph-centric view of dialogue management, using either handcrafted rules or data-driven models. We also briefly discuss how to account for some aspects of dialogue management such as uncertainties, incremental inputs and contextual knowledge. Finally, we describe a proof-of-concept study of this dialogue management framework in a human-robot interaction scenario.

1 Introduction

Representing, updating and acting upon the current dialogue state is at the core of dialogue management. For task-oriented systems, this dialogue state is often repre-

Nicholas Walker
Norwegian Computing Center, Postboks 114, Blindern 0314 Oslo, e-mail: walker@nr.no

Torbjørn Dahl
Department of Informatics, University of Oslo e-mail: torbjd@ifi.uio.no

Pierre Lison
Norwegian Computing Center, Postboks 114, Blindern 0314 Oslo, e-mail: plison@nr.no

sented by a fixed, predefined list of *slots* to fill [19, 16]. However, this representation in terms of slot-value pairs is difficult to apply to open-ended domains with varying numbers of entities to track. For example, human–robot interaction tasks must often keep track of entities such as locations, persons or tasks to perform. Those entities may vary over time – for instance, the number of persons located in a given room is not known in advance and may change over the course of the interaction. Those entities are also connected with one another through various relations, such as a person being *in* a room, or a response being an *answer* to a preceding utterance.

Graphs are well suited to represent such rich relational structures between (abstract or concrete) entities. Graphs, and machine learning models operating on those graphs (in particular *graph neural networks*) are also increasingly popular for dialogue modelling [6, 9, 5] and generally in NLP [20]. However, such approaches generally focus on specific dialogue modelling aspects and eschew the more general question of how to design a full-fledged dialogue manager operating on a dialogue state expressed as a graph. This paper is a first attempt at answering this question. We describe how to (1) encode the dialogue state as a *property graph* and (2) frame dialogue management as sequences of *graph transformations* that iteratively refine this state (and select actions to perform) based on incoming observations.

We also show how these manipulations can be expressed in a graph query language called OpenCypher [7] and executed on a graph database. Using a unified, declarative language for all dialogue operations allows us to clearly separate the domain-specific logic (which graph operations to execute and on the basis of which inputs) from implementation issues related to e.g. concurrency and query optimisation (which are handled by the back-end graph database). It also makes it possible to query knowledge graphs (expressing background knowledge) using the same syntax.

The proposed approach aims to accommodate both handcrafted rules and machine learning models. This ability to combine rule-based and data-driven modules is important when operating on rich, graph-based state representations, as the complexity of the resulting state-action space makes it difficult to learn end-to-end models, at least for domains without large amounts of training data readily available.

We start by briefly reviewing related work on graph-based dialogue management (Section 2), then sketch our dialogue management approach, with a particular focus on the representation of the dialogue state and formalisation of dialogue management tasks with graph transformations (Section 3). Section 4 illustrates this framework with a case study in human–robot interaction, and Section 5 concludes.

2 Related Work

The use of graphs – or more generally, relational representations – in dialogue management has been explored in several previous works. Earlier rule-based approaches to dialogue management often relied on rich formalisations of the dialogue state encoding the beliefs, desires and intentions of each conversational partner through logical forms [2, 12, 11]. However, such formalisations are typically limited to high-

level symbolic knowledge, thereby leaving out non-symbolic information such as spatio-temporal features from the dialogue state. Incremental approaches to dialogue processing [17] also rely on relational representations to connect incremental units with one another through temporal or semantic links.

Graphs have also been used as part of statistical and neural conversational models. One important instance is the use of probabilistic graphical models such as Bayesian Networks [18, 14]. However, the "relations" defined in such models are limited to conditional probability distributions between random variables, and cannot as such express other, more semantic relationships. Graph neural networks have also been employed for dialogue policy learning of slot-based systems, such as in [6].

A number of papers have also focused on the use of knowledge graphs to improve the quality of dialogue responses. A sequence-to-sequence conversational model relying on graph embeddings derived from a knowledge graph is presented in [10]. In [8], the authors present a knowledge-grounded conversational model that exploits a large knowledge graph to derive more content-rich responses to user queries. The authors of [21, 13] make use of a graph-encoded knowledge base to inform a dialogue system along with dependency parses of sentences. Finally, [15] show how to integrate a graph database (expressed as RDF triples) to a social chatbot.

Graph representations are also a core element of conversational semantic parsing, although the graphs are here limited to relations within a given utterance and do not typically cross utterance boundaries – although see [3] for an exception that allows for some semantic relations (references, repairs) across utterances.

The main novelty of the proposed approach is its reliance on a unified graph to track all variables relevant to dialogue management (including e.g. utterances, speakers, entity mentions, conversational intents, external knowledge, etc.). Dialogue state tracking and action selection are then framed as operations that continuously manipulate this graph to incorporate incoming observations and select new system actions. Those updates are expressed in a declarative language and run on a graph database, making it possible to handle concurrent read/write operations and allow for arbitrarily complex manipulations of the state graph.

3 Approach

The system architecture (illustrated in Fig. 1) revolves around a blackboard design where various software modules continuously listen for changes (insertion, deletion or modification of nodes or edges) to the central dialogue graph and generates further updates to this graph whenever necessary. All updates are specified through graph queries encoded in OpenCypher [7]. The dialogue state itself is stored using an in-memory graph database[1]. The architecture also supports message-passing with ZeroMQ [1] to allow modules to exchange data that are not relevant for dialogue management and do not need to be inserted into the dialogue state.

[1] See `http://www.memgraph.com`

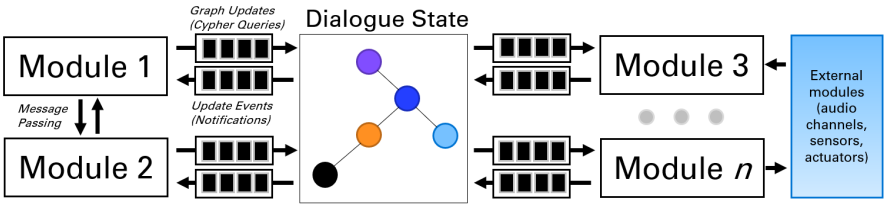

Fig. 1: General system architecture, with the graph database storing the current dialogue state at its centre. Each dialogue management module is notified of new updates to the dialogue state, and may itself submit further updates (expressed as graph queries). Modules may also receive/submit data to one another through message-passing. All modules run in parallel and have both an input queue (of update events to process) and an output queue (of graph queries introducing further changes to the state), thereby allowing for asynchronous processing.

Modules can be easily plugged in and out of the architecture, and may correspond to e.g. handcrafted rules or data-driven models. Note that there is no explicit distinction between dialogue state tracking and action selection – both operations are expressed using the same graph manipulation mechanisms.

3.1 Dialogue state

We model the dialogue state as a *property graph* [7], which is graph structure allowing both nodes and edges to be associated with properties and labels. Property graphs are often contrasted with *triple stores* such as RDF, which cannot directly attach properties to nodes or edges without having to create new entities.

We require each node and edge to have a semantic label such as `Utterance`, `Speaker`, `Intent` or `Location`. Those nodes may represent observable entities but may also express abstract objects such as a task to perform. The labels attached to each node allow modules to directly filter state updates (for instance, the insertion of a new `Intent` will trigger subsequent updates related to action selection). The architecture does not impose any particular set of labels. However, we do rely on a number of conventions to account for important dialogue modelling aspects:

Uncertainty: Accounting for uncertainties and partial observability is an important consideration in dialogue systems, especially in domains such as human–robot interaction where observations are often noisy or error-prone. We express discrete distributions by inserting a central node denoting the random variable itself and a set of outgoing nodes attached through a special `ALTERNATIVE` relation, each node corresponding to a distinct value and associated probability. Discrete conditional distributions between two random variables can be similarly expressed through edges indicating the conditional probability between two

values. This representation can only express a limited form of probabilistic knowledge – in particular, it does not account for uncertainties related to edges, nor does it capture continuous distributions or conditional distributions with more than one independent variable. Nevertheless, this representation can express most common forms of uncertainties in dialogue management, such as N-Best lists and the probabilistic outputs of machine learning models.

Temporality: Time is a crucial aspect of dialogue management, in particular to implement flexible turn-taking strategies. To this end, we treat time as a core component of the graph and associate each node with timestamps expressing its time of creation and its last update. Entities with a duration (such as `Utterances`) also include start and end timestamps. This temporal information makes it possible to 1) explicitly reason over temporal aspects of the interaction and 2) analyse how the dialogue state evolves over time.

Incrementality: As argued in e.g. [17], human speakers process dialogues *incrementally*, by gradually refining their interpretation of what is being said (and producing appropriate responses) on the basis of small units of content. To emulate such a behaviour in a dialogue system, one needs the ability to chain together such small units and revise/revoke some of these units whenever necessary. Incremental content can be expressed in our framework through a special `PREVIOUS` relation connecting together consecutive units, and be revoked by deleting the content along with all nodes derived from it[2].

Contextual knowledge: Finally, dialogue systems often need to access background knowledge to fulfil their tasks. One important benefit of graph-centric dialogue management is the fact that such background knowledge can often be conveniently encoded as a knowledge graph and be queried using the same syntax as other dialogue management operations (as shown in our case study).

3.2 Graph operations

Each dialogue management module listens for notifications of changes in the dialogue state and (when necessary) outputs further updates in the form of graph queries (see Fig 1). Those modules can be implemented in several ways:

1. The simplest method is to write a graph query associating a given condition to a state update. For instance, the rule below specifies that, if an utterance mentions an entity named x and our knowledge graph includes a person whose full name starts with x, a `REFERS_TO` edge can be created between the two[3]:

```
MATCH (mention:EntityMention), (person_in_kb:Person)
WHERE person_in_kb.name STARTS WITH mention.name
CREATE (mention)-[:REFERS_TO]->(person_in_kb);
```

[2] This functionality is, however, limited to incremental units with a relatively modest throughput, and is not appropriate for handling high-frequency events (as is the case for e.g. audio data).

[3] For simplicity, we ignore here how to handle ambiguous references with multiple potential targets.

Expressing state updates directly through graph queries allows us to leverage the expressive power of the Cypher language to detect complex graph patterns.
2. Alternatively, one can produce graph updates directly through Python code. Each module has read access to the dialogue state (again through graph queries executed onto the current dialogue state) to extract the inputs necessary for inference, and outputs a list of update queries in the form of `CREATE`, `MERGE`, `SET` or `DELETE` commands. Modules can notably take advantage of machine learning models and output probabilistic predictions that are then used to update the graph.

4 Case study

We used a simple human–robot interaction scenario to showcase how the proposed dialogue management framework can be applied in practice. The robot objective was to function as an automated receptionist, and more specifically (1) answer questions related to the availability of various researchers as well as (2) accompany visitors to a few selected places on the current office floor.

We relied on a knowledge graph storing the calendar data of all researchers to answer questions related to the whereabouts of each person. We use a Pepper robot as platform, along with Google Speech for speech recognition and the TTS engine embedded in Pepper. For NLU, we used a neural intent classifier and entity extractor with a pretrained model from Rasa [4] fine-tuned with a small list of domain-specific examples. Once a new `Intent` is added to the graph, the rest of the dialogue management process is implemented through graph queries.

A step-by-step example of such process is illustrated in Fig. 2. Due to space constraints, we only provide a high-level description of each update, but the detailed list of graph queries employed to perform each operation is available at:
https://github.com/NorskRegnesentral/GraphDial.

5 Conclusion

This short paper presented ongoing work on a novel, graph-centric approach to dialogue management in which dialogue state tracking and action selection are viewed as *graph manipulation problems*. The dialogue state is represented as a property graph. Our case study demonstrates the utility and feasibility of a graph-centric dialogue management system in a human–robot interaction setting.

Along with the further development of the system architecture (and its release as an open-source toolkit), future work will concentrate on scaling up the case study with a larger set of intents and system responses, and on conducting a proper evaluation of the resulting platform. We also aim to investigate how to integrate graph neural networks into the architecture, as those types of neural models are ideally suited to exploit the relational structure expressed in the state graph.

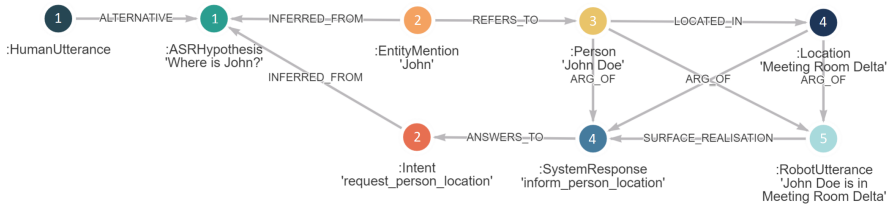

Fig. 2: Illustration of graph operations upon a new user utterance "Where is John?"

Step 1: The speech recogniser recognises a new utterances and inserts in the dialogue graph one new `HumanUtterance` node (with various information such as timestamps etc.), along with one `ASRHypothesis` node attached to it.

Step 2: The NLU module is notified of this utterance and classifies it as a new request_person_location intent, along with the person mention "John".

Step 3: The mention "John" is connected to a person entity "John Doe" in the knowledge graph through a simple reference resolution rule (see Section 3.2)

Step 4: Another rule detects the presence of a request_person_location intent connected to a person with a known location, and produces inform_person_location as possible response with high utility. This response is then selected by another rule selecting the response with highest utility among possible candidates.

Step 5: This response triggers the creation of a `RobotUtterance` with the `Person` and `Location` as arguments. This utterance is picked up by speech synthesis.

References

[1] Faruk Akgul. 2013. *ZeroMQ*. Packt Publishing.

[2] James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. 2000. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6(3–4):213–228.

[3] Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, et al. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

[4] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *CoRR*, abs/1712.05181.

[5] Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *Proc. of AAAI*, volume 34, pages 7521–7528.

[6] Lu Chen, Bowen Tan, Sishan Long, and Kai Yu. 2018. Structured dialogue policy with graph neural networks. In *Proc. 27th International Conference on Computational Linguistics*, pages 1257–1268.

[7] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer,

and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proc. 2018 International Conference on Management of Data*, SIGMOD '18, page 1433–1445. Association for Computing Machinery.

[8] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proc. 32nd AAAI Conference on Artificial Intelligence*, pages 5110–5117.

[9] Deepanway Ghosal, Navonil Majumder, Soujanya Poria, Niyati Chhaya, and Alexander Gelbukh. 2019. DialogueGCN: A graph convolutional neural network for emotion recognition in conversation. In *Proc. EMNLP-IJCNLP*, pages 154–164.

[10] He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proc. 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776.

[11] Kristiina Jokinen. 2009. *Constructive Dialogue Modelling: Speech Interaction and Rational Agents*. Wiley-Interscience.

[12] S. Larsson and D. R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3-4):323–340.

[13] Lizi Liao, Le Hong Long, Yunshan Ma, Wenqiang Lei, and Tat-Seng Chua. 2021. Dialogue state tracking with incremental reasoning. *Transactions of the Association for Computational Linguistics*, 9:557–569.

[14] Pierre Lison. 2015. A hybrid approach to dialogue management based on probabilistic rules. *Computer Speech & Language*, 34(1):232 – 255.

[15] Jan Pichl, CTU FEE, Petr Marek, Jakub Konrád, Martin Matulík, Jan Šedivỳ, and CTU CIIRC. 2018. Alquist 2.0: Alexa prize socialbot based on sub-dialogue models. *Alexa Prize Proceedings*.

[16] Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *Proc. 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2786.

[17] David Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue & Discourse*, 2(1):83–111.

[18] Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562.

[19] Jason D Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

[20] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090*.

[21] Shiquan Yang, Rui Zhang, and Sarah Erfani. 2020. Graphdialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems. In *Proc. 2020 Conference on Empirical Methods in Natural Language Processing*, pages 1878–1888.