Formal Dialogue and Large Language Models

Mark Snaith[1,*,†], Simon Wells[2,†]

[1]School of Computing, Engineering and Technology, Robert Gordon University, Aberdeen, AB10 7GJ, Scotland, UK
[2]Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, Scotland, UK

Abstract
In this paper, we present preliminary work into combining formal models of dialogue and large language models, before going on to discuss how this provides a foundation for similar approaches involving computational models of argument. First, we address the twin issues of how a formal dialogue game can usefully regulate dialogical utterances generated by an LLM during an extended, goal-oriented conversation, and conversely, how LLMs can close the human-level language generation gap associated with formal dialogue games. We then proceed to identify how our solution to these issues can underpin future work towards using computational argumentation to provide reasoning-like capabilities to LLMs, and using LLMs for tasks such as argument mining, and searching and summarisation of analysed argument data.

Keywords
Formal dialogue, argumentation,

1. Introduction

In this paper, we first address the twin issues of how a formal dialogue game can usefully regulate dialogical utterances generated by an LLM during an extended, goal-oriented conversation, and conversely, how LLMs can close the human-level language generation gap associated with formal dialogue games. To that end, we present the PreFACE library, a tool that allows a software agent to query an LLM for an appropriate response given their currently available dialogue move(s). We then proceed to identify further potential uses for PreFACE in underpinning future work towards using computational argumentation to provide reasoning-like capabilities to LLMs, and using LLMs for tasks such as argument mining, and searching and summarisation of analysed argument data.

In [1] the authors addressed questions about the role of formal dialogue in the age of LLMs and established that (1) LLMs, at least in their current form, do not yet subsume all of the functionalities of formal approaches to dialogue, and (2) that formal dialogue games can have an important regulatory role, ensuring that the dialogues that are generated conform to normative expectations of how dialogues should progress. Whilst it might be the case that future generations of LLM will effectively engage in focused and extended argumentative dialogues of various types, there are situations where it is unfeasible to continuously retrain an LLM, or to supply all of the domain and corpus knowledge to the LLM, especially where interacting agents are heterogeneous, owned, operated or governed by differing individuals and organisations, and maintaining proprietary knowledge that is outside of the LLM.

2. Background

Research into dialogue games originated in a variety of goal oriented studies; whether to explain proof procedures [2], to model the Aristotelian conception of Dialectic [3], to understand how people interact for example, during deliberation [4], or to manage the interactions between intelligent agents in multi-agent systems [5]. At various times, researchers have attempted to determine ways to organise these

SICSA Workshop on Reasoning, Explanations and Applications of Large Language Models, October 17th 2024, Aberdeen, Scotland
*Corresponding author.
†These authors contributed equally.
m.snaith@rgu.ac.uk (M. Snaith); s.wells@napier.ac.uk (S. Wells)
https://www3.rgu.ac.uk/dmstaff/snaith-mark/ (M. Snaith); https://www.simonwells.org/ (S. Wells)
0000-0001-9979-9374 (M. Snaith); 0000-0003-4512-7868 (S. Wells)

various approaches. For example, McBurney and Parsons [6] proposed a set of desiderata associated with dialogue games in agent communication, similarly, Wells [7] put forward a number of criteria for specifying dialectical games which was subsequently developed into the Dialogue Game Description Language [8]. These approaches were all broadly in the context of intelligent agent interaction via dialogue games, and mainly sought to give structure within which benchmarks, standards, expectations, and points of comparison, could be situated whilst also seeking to depict the space of possible dialogue games so that a principled exploration could be made.

Various tools and platforms have been developed to support computational implementations of dialogue games, and their subsequent execution to be used in inter-agent communication. The Dialogue Game Description Language (DGDL) is a domain-specific language for describing dialogue games [8], while the Dialogue Game Execution Platform (DGEP) provides an environment for interpreting and running games specified in DGDL. Together, DGDL and DGEP have been shown to support systems for public deliberation [9] and health coaching [10], as well as underpinning generalised platforms for structured conversational systems [11, 12] and influencing related work in this area [13].

By contrast, Large Language Models (LLMs) provide a less structured approach to conversational interaction. Given a prompt, they will predict the best response to make. Prompt engineering is the process of designing and structuring prompts towards more effective results. This leverages an LLMs ability for in-context learning [14], where in addition to its underlying model, the LLM temporarily learns either from the prompt itself, or via information provided specifically as context from which it should derive its response. One approach to the latter is Retrieval Augmented Generation (RAG) [15]. A RAG-based system first retrieves a set of documents (using semantic searching), then feeds those documents into the LLM as context for producing the final output.

The differences between dialogue games and LLMs are therefore quite clear: the former provide a structured account of how a multi-party, goal-oriented conversation should proceed, without any consideration as to the precise content of each move. LLMs, on the other hand, are adept at generating natural language text in response to arbitrary prompts, without any consideration as to "legal" dialogical (conversational) flow. These differences, however, present opportunities for each to enhance and support the capabilities of the other. Combining the rigid dialectical structures provided by dialogue games, with the rich generative capabilities of LLMs will simultaneously address the human-level language gap associated with the former, and the lack of useful regulation of utterances associated with the latter.

3. The PrEFACE Library

The PrEFACE (Prompt Engineering for Argumentative Conversational Exchanges) library is a tool for closing the loop between Formal Dialogues Games and Large Language Models. Figure 1 illustrates PrEFACE schematically from an Input/Output perspective. PrEFACE is designed to sit between an LLM and an agent such that the agent decides what kind of utterances to make, based upon the current dialogical context, and makes a request to PrEFACE, supplying the requisite dialogue context request structure. PrEFACE uses this structure to generate a prompt which is then provided to the LLM. The LLM generates a response, which is returned to PrEFACE. The response is then returned to the agent alongside additional metadata. The following sub-sections describe each stage of the process in more detail.

3.1. The Dialogical Context Request Structure (DCRS)

The Dialogical Context Request Structure (DCRS) is the primary means by which PrEFACE is directed to generate prompts. A completed DCRS is a JSON[1] document that describes the current dialogical context along with the associated utterance that it is requested for the LLM to generate. The DCRS is shown in figure 2 and corresponds to stage 1 of figure 1 PrEFACE builds on the Dialogue Game Description Language (DGDL) [8] as a pragmatic initial tool for consistently describing the moves within a dialogue.

[1]http://www.json.org/

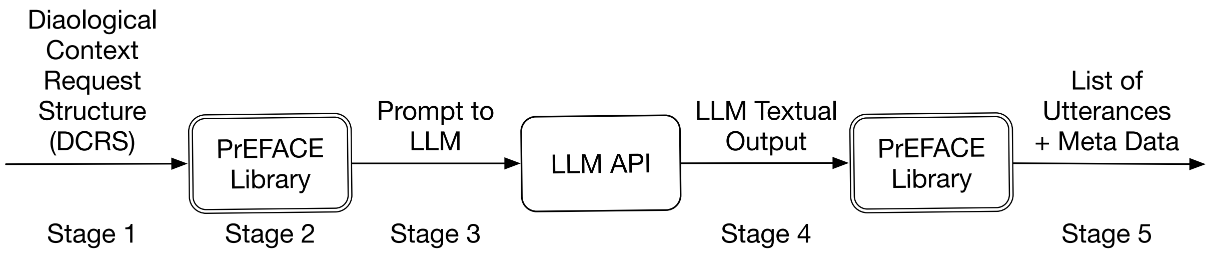

Figure 1: The stages of activity during PrEFACE based prompt engineering.

```
{
    "metadata": {
        "name": "dcrs",
        "version": "1.0"
    },
    "history": {
        "utterances": [
            {
                "index": "",
                "speaker": "",
                "move_name": "",
                "content": ""
            }
        ]
    },
    "topic": {
        "description": "",
        "stance": ""
    },
    "response": {
        "move_name": "",
        "scaffold": "",
        "opener": "",
        "requirements": [],
        "effects": []
    },
    "knowledge": {
        "type": "",
        "content": []
    }
}
```

Figure 2: The Dialogue Context Request Structure JSON format

Because DGDL has a finite range of components that are used to describe each locution, there are a finite number of prompt templates that need to be constructed within PrEFACE.

The DCRS is an object comprising five blocks, including metadata, history, topic, response, and knowledge. A completed DCRS object, in JSON format, is supplied to PrEFACE to initiate the prompt generation process.

The first block, metadata, is necessary to positively identify an instance of the DCRS so that it can be verified against the correct version of the DCRS JSON schema. This block is mandatory but after schema verification, plays no further part in prompt generation.

The second block, history, provides a list of utterance objects that correspond to prior interactions during the dialogue that are relevant to the utterance to be generated. Each utterance object comprises an index, speaker, move_name, and content. The index key is negatively indexed from the current utterance, the one being generated, e.g. the immediate previous utterance was -1, the one before that was -2, etc. Under most circumstances, an utterance will be generated to respond to the last utterance of another participant to the dialogue. In a small number of cases, two or more previous utterances might be required in order to provide additional context for generation of the target utterance, for example, to give the context of a micro-exchange within the dialogue, e.g. a Question-Answer-Challenge complex. A third scenario occurs when backtracking occurs within the dialogue, returning to address an utterance that had previously been made within the dialogue and was perhaps insufficiently resolved at that earlier time-point. Speaker refers to the dialogue participant that uttered this content. "self" is used to indicate that an utterance was communicated by the current speaker who is generating the PrEFACE utterance. Speaker identification is required to differentiate between the current speaker, the participant to whom they are responding directly, i.e. to a question directed from another participant towards the current speaker, and also to differentiate any other participants in the dialogue who might have previously participated in the current micro-dialogue. For example, speakers A, B, and C where speaker A poses a question, speaker B (self) answers it, and speaker C then challenges speaker B"s answer, leading to self making a defence oriented towards speaker C but directly related to the originating question from speaker A. The move_name key refers to the DGDL move_name label, used to distinguish between available moves. Finally content refers to the propositional content of the move, expressed as a string.

The response block is used to specify the kind of utterance the agent requires the LLM to generate. For any given move within a DGDL game description there may be a range of either prescribed or permitted responses which together constitute the set of legal moves. PrEFACE is designed to handle one move at a time, so an agent must process the legal moves list one element at a time, passing in the specific move type that must be generated. This way PrEFACE is focused upon engineering a single prompt per dialogue context and where there are legal moves that could be generated, but which fall outside of the agents strategy, resources are not wasted. Responses comprise a move_name, scaffold, opener, requirements, and effects. The move_name is a string describing the required move which would usually constitute a DGDL move_name or a speech act label. The scaffold is a string template that defines the form of the required move, for example. “It is not the case that X because Y”, where X and Y are parameters that must be filled when generating an utterance. Not all dialogue games specify a scaffold for their moves however. More common is the opener, as found in the Ravenscroft’s Critical Reasoning Game (CRG)[16] where moves such as ‘Suggest1’ specifies the opener “My idea is”. Completing the response block are the requirements and effects, which directly map onto the equivalent elements of the DGDL specification for the associated move.

The Topic block is used to communicate an overall topic for the dialogue and stance regarding that topic, to the LLM. The topic is a string description and the stance is a single string from the set of {“support” | “neutral” | “oppose” }.

Finally, the knowledge block is used to specify a subset of the agent’s domain knowledge that can assist the LLM in generating higher quality utterances. The DCRS uses information, e.g. descriptions of moves and knowledge base (KB) contents, to provide context that is both relevant, only the move sequence that needs to be responded to is supplied, and efficient, only the subset of knowledge that the agent deems relevant to this exchange is used in the subsequent prompt generation. This means that the calling agent, the one providing the DCRS, must decide which knowledge is relevant to supply, and which locution to request in response, from the space of legal moves as defined by the dialogue game. Where there are multiple possible responses that the agent might make, then the agent must make multiple requests via PrEFACE.

Whilst an LLM encodes a huge amount of information, this is generic, and often conflicting, information, due to the wide ranging nature of the training sets used to create the model. In contrast, an intelligent agent likely has a KB that is both coherent and specific to its use case. The knowledge block is used, optionally, to supply sufficient, specific, knowledge to the LLM to enable the prompt to tune the LLMs’s response. Conversely, there are two cases in which an agent might opt to not

```
You are assisting a user in a dialogue on the topic of: {topic.description}.
Their stance is: {topic.stance}.

Your job is to find a value for $response consistent with $knowledgebase.
Use $context to help but don't repeat anything contained within it

$knowledgebase = [{knowledge.content}]}

$context = [{history.utterance}]}

Return only a response without any extra text
```

Figure 3: Example context prompt

supply a knowledge block to PrEFACE. The first is when considering the minimal case for generating an utterance using a DGDL fragment containing an opener and a stance and the second is when the relevant knowledge within an agent's KB has been fully explored, but the agent is not yet ready to capitulate the dialogue, in this case the LLM might return a serendipitously useful utterance that enables the dialogue to continue based upon the generalised knowledge encoded in the LLM.

3.2. Prompt generation

Given a DCRS document, PrEFACE uses the attributes to generate a suitable prompt that will, at least in principle, return an appropriate response to be used as content for the dialogue move. The primary basis of this prompt is the *response.scaffold* field, with the LLM instructed to find a value for the missing parameter(s), however other attributes also play a role in ensuring a relevant response.

In Section 3.1, we describe how the DCRS contains *history* and *knowledge* blocks. The content of these are provided as context to the LLM, along with instructions that define the role the LLM should fulfil. Our approach to crafting this context is based on OpenAI's *Assistants API*[2]. An Assistant is designed to follow specific instructions and make use of provided contextual information in responding to user queries.

The structure of the context has been designed to encode 1) the topic and stand; 2) the specific task and associated constraints; 3) the knowledge and history provided as part of the DCRS; and 4) a specific instruction to provide only the response without extraneous text. An example of this structure is shown in Figure 3.

The prompt itself is then based on the value provided in *response.scaffold.* This scaffold, defined by the game developer alongside the DGDL specification, provides a natural language template for what form the content should take in fulfilling the move. As an example, the scaffolding for an *argue* move may be *"$p because $q"*, where $p is a variable that will be instantiated with some previous claim, and $q is a variable that will ultimately be sourced from the LLM. Section 3.3 provides worked example showing how the scaffold becomes a concrete prompt.

3.3. Worked Example

A request to PrEFACE consists of six steps, from the point at which the dialogue machinery indicates that the agent has available moves, through to those moves being instantiated with propositional content. Here we provide a worked example of the full process.

We assume that a dialogue is taking place, based on a DGDL description that is being executed by the Dialogue Game Execution Engine (DGEP), a platform for running and regulating implemented formal dialogue games. The versions of DGDL and DGEP that we use are those incorporated into the Agents

[2]https://platform.openai.com/docs/assistants/overview

```
{
    "agent":[
        {
            "moveID":"argue",
            "opener":"because $q",
            "target":"human",
            "reply":{
                "p": "We should take climate change seriously",
                "q": "$q"
            }
        },
        ...
    ]
}
```

Figure 4: JSON excerpt of available moves from DGEP

```
{
    "speaker":"agent",
    "target":"human",
    "reply":{
        "p": "We should take climate change seriously",
        "q": "Ignoring it would have devastating consequences"
    }
}
```

Figure 5: JSON format of a reply to be sent to DGEP

United platform[3] [12], which returns the set of available moves as a JSON object in the form shown in Figure 4. Where DGEP identifies that the agent has available moves, the PrEFACE request process begins. While all moves are subject to the process, for brevity we focus on a single move, $argue$.

For each available move, the first stage is to create the concrete DCRS object. This involves adding relevant utterances and knowledge to the $history$ and $knowledge$ fields respectively, and generating a concrete scaffold that will be used as the basis of the LLM prompt. The DCRS for an $argue$ move contains a scaffold of the form *\$p because \$q*, where *\$p* and *\$q* are variables representing the two components of the argument. The value for *\$p* is obtained from the $reply$ object in the move, while the value for *\$q* is the response to be sourced from the LLM. Our concrete scaffold therefore becomes *We should take climate change seriously because \$response.*

The second stage is to initialise PrEFACE using the DCRS, which in turn sets up the necessary objects and functions for interacting with the LLM, and generates the final prompt that will be submitted. This includes providing the instructions and context described in section 3.2. One PrEFACE is initialised, the fourth stage is to send the prompt and await the response. To ensure a tightly-bound response, we leverage the *function calling* capabilities of GPT-based LLMs, with the result passed to a callback function whose eventual purpose will be to verify the response[4].

When a final response is obtained, the fifth stage is to embed that response into the reply that - assuming this move is chosen - will be returned to DGEP. This reply is also a JSON object, of the form shown in Figure 5. The sixth and final stage is to submit this object to DGEP, which will then determine the next available moves and, if any are for the agent, start the process again.

[3]Available from https://github.com/AgentsUnited
[4]It is as-yet unclear what such a verification would involve, and so we leave this to future work.

4. Future developments and directions

The version of the PreFACE library described in section 3 provides a simple method of linking formal models of dialogue with large language models. This however represents only an initial starting point; PreFACE remains under active development, not only to enhance its capabilities in a dialogical context but also as a generalised tool for harnessing the capabilities of LLMs in the broader field of computational argumentation. In this section, we briefly discuss future developments, from the immediate term to a longer-term vision.

4.1. Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) allows LLMs to use external sources in generating responses to prompts. A common use-case is allowing an LLM to access domain-specific documents as a way of providing conversational interfaces for users to find relevant information (e.g. an employee asking *"What is the company annual leave policy?"*, and the RAG-supported LLM will retrieve the answer from the relevant document). Within PreFACE, RAG provides opportunities to further supplement an agent's knowledge through retrieval of relevant arguments from a data store such as ArgDB[5] [17]. A vector-based version of ArgDB is currently under development, and we intend to use this to integrate RAG capabilities into PreFACE in the near future.

There are however further hurdles that will need to be overcome. ArgDB stores analysed argument data as directed graphs, represented in JSON. While LLMs are capable of parsing and querying JSON, they are not as yet capable of understanding the semantics; that is, they cannot understand the inference and conflict relationships described by the JSON objects. An additional step is therefore required to represent the JSON returned from ArgDB in a format that can more readily be interpreted by an LLM.

4.2. Argument Summarisation

As noted in Section 4.1 above, analysed argument data is stored as directed graphs. These graphs comprise individual premises, conclusions and the relationships between them, and can be easily visualised. Generating textual summaries of these analyses can be useful, for example in helping understand a complex topic with multiple conflicting viewpoints.

It is our intention in future work to explore how PreFACE can be extended to not only find suitable content for a dialogue move, but also provide summaries of analysed arguments stored in ArgDB. We envisage that this will leverage the (upcoming) RAG capabilities, but instead of finding a specific piece of content to fulfil a dialogical function, instead the LLM will be used to summarise a collection of arguments and the relationships between them.

5. Conclusions

This paper has presented preliminary and in-progress work towards combining the strict dialectical structures imposed by formal dialogue games, and the human-level language generation capabilities of large language models (LLMs). We presented the PreFACE library, a tool that allows a software agent to query an LLM for an appropriate response given their currently available dialogue move(s). The Dialogue Context Request Structure (DCRS) allows the agent to provide the current dialogical context along with other associated details relevant to the utterance that the LLM is being request to generate.

As the capabilities of LLMs continue to expand, so to will the demand for further application areas. The work we present here lays a foundation to make such expansions into domains that require both strong natural language generation, and strict conversational structures.

[5]https://github.com/Open-Argumentation/ArgDB

References

[1] S. Wells, M. Snaith, On the role of dialogue models in the age of large language models, in: Proceedings of the the 23rd International Workshop on Computational Models of Natural Argument (CMNA'23), 2023.

[2] P. Lorenzen, K. Lorenz, Dialogische Logik, Dormstatdt, Wissenschftliche Buchgesellschaft, 1978.

[3] N. Rescher, Dialectics, State University of New York Press, Albany, 1977.

[4] D. Godden, S. Wells, Burdens of proposing: On the burden of proof in deliberation dialogues, Informal Logic 42 (2022) 291–342. URL: https://informallogic.ca/index.php/informal_logic/article/view/7225/5379.

[5] S. Wells, Formal Dialectical Games in Multiagent Argumentation, Ph.D. thesis, University of Dundee, 2007.

[6] P. McBurney, S. Parsons, M. Wooldridge, Desiderata for agent argumentation protocols, Proceedings of the First AAMAS (2002) 402–409. URL: http://www.csc.liv.ac.uk/~peter/downloads/aamas02.ps.

[7] S. Wells, C. Reed, Testing formal dialectic, in: Proceedings of the Second International Workshop on Argumentation in Multi-Agent Systems (ArgMAS), 2006.

[8] S. Wells, C. Reed, A domain specific language for describing diverse systems of dialogue, Journal of Applied Logic 10 (2012) 309–329.

[9] M. Snaith, J. Lawrence, C. Reed, Mixed initiative argument in public deliberation, in: F. De Cindio, A. Macintosh, C. Peraboni (Eds.), From e-Participation to Online Deliberation, Proceedings of the Fourth International Conference on Online Deliberation, OD2010, Leeds, UK, 2010, pp. 2–13.

[10] M. Snaith, H. op den Akker, T. Beinema, M. Bruijnes, A. Fides-Valero, G. Huizing, R. Kantharaju, R. Klaassen, K. Konsolakis, D. Reidsma, M. Weusthof, A demonstration of multi-party dialogue using virtual coaches: the first council of coaches demonstrator, in: Proceedings of the 7th International Conference on Computational Models of Argument (COMMA 2018), Warsaw, Poland, 2018, pp. 473–474.

[11] R. B. Kantharaju, A. Pease, D. Reidsma, C. Pelachaud, M. Snaith, M. Bruijnes, R. Klaassen, T. Beinema, G. Huizing, D. Simonetti, D. Heylen, H. op den Akker, Integrating argumentation with social conversation between multiple virtual coaches, in: IVA 2019 - Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents, Association for Computing Machinery, Paris, France, 2019, pp. 203–205.

[12] T. Beinema, D. Davison, D. Reidsma, O. Banos, M. Bruijnes, B. Donval, Á. F. Valero, D. Heylen, D. Hofs, G. Huizing, R. B. Kantharaju, R. Klaassen, J. Kolkmeier, K. Konsolakis, A. Pease, C. Pelachaud, D. Simonetti, M. Snaith, V. Traver, J. Van Loon, M. H. Visser, Jacky Weusthof, F. Yunus, H. J. Hermens, H. op den Akker, Agents united: An open platform for multi-agent conversational systems, in: Proceedings of the 21st ACM International Conference on Intelligent Virtual Agents, 2021, pp. 17–24.

[13] T. Krauthoff, M. Baurmann, G. Betz, M. Mauve, Dialog-based online argumentation., in: COMMA, 2016, pp. 33–40.

[14] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al., Emergent abilities of large language models, arXiv preprint arXiv:2206.07682 (2022).

[15] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[16] A. Ravenscroft, S. Wells, M. Sagar, C. Reed, Mapping persuasive dialogue games onto argumentation structures, in: AISB Symposium on Persuasive Technology & Digital Behaviour Interventions, 2009.

[17] S. Wells, Datastores for argumentation data., in: CMNA@ COMMA, 2020, pp. 31–40.