Dialogue grammar induction

MARK SNAITH
Centre for Argument Technology, University of Dundee, UK
m.snaith@dundee.ac.uk

CHRIS REED
Centre for Argument Technology, University of Dundee, UK
c.a.reed@dundee.ac.uk

This paper presents a foundation for inducing formal dialogue games from analysed transcripts of real, inter-human conversations. We describe the *DI-Algorithm* (Dialogue Induction Algorithm), that accepts as input transcripts analysed using the Argument Interchange Format (AIF) enriched with Inference Anchoring Theory (IAT), from which it induces a formal, context-free grammar. This grammar describes the dialogue protocol that was followed in order to generate the original transcript. To illustrate the *DI-Algorithm*'s application, we provide a worked example based on an AIF analysis of a mock conversation.

KEYWORDS: dialogue games, dialogue protocols, grammar induction

1. INTRODUCTION

Philosophical dialogue games, such as those proposed by Hamblin (1970), Walton (1984) and Walton and Krabbe (1995) have been used to influence computational protocols for argumentative inter-agent communication (Reed, 1998; McBurney & Parsons, 2002a, 2002b; Black & Hunter, 2009). These games have traditionally been specified by hand. This, however, is a process that has two significant drawbacks; first, for all but the most trivial games it is time-consuming to generate the rules that define the protocol and account for every possible situation; second, it is rare for dialogue game specifications to accurately reflect the way in which real (human) dialogues progress. This second drawback is particularly relevant to the recent emergence of mixed-initiative argumentation (Snaith et al., 2010), where the participants in a dialogue can be real, virtual or a mixture of both. For users to remain engaged in a mixed-initiative dialogue governed by a formal protocol, it is imperative that permitted moves allow the dialogue to advance in a natural way, especially in domain-specific scenarios. If we are to model such natural dialogue flow in an artificial environment, it is essential that we understand the rules and protocols that are (in many cases subconsciously) followed in real, everyday dialogues.

Types of natural dialogue vary immensely in terms of both structure and the rules that govern their flow; contrast, for instance, a highly strict and formal courtroom setting, with some friends chatting over coffee. In the former, the rules are largely explicit with all parties involved strictly abiding by what they can say and when. In the latter, the structure is much more casual and simply flows based on what has previously been said. Nevertheless, even in formal settings there are certain implicit rules and norms that regulate the conversation (e.g. it is rude to interrupt, you should not suddenly change the topic, questions should typically be answered etc.).

One method of gaining an understanding of the structure of natural dialogues in a given domain or setting is to analyse transcripts of the conversations those dialogues generate. Discourse analysis (or mapping) involves breaking down individual components of an argument or other linguistic structure and re-assembling them in a form that shows the relationship between individual claims and premises (Van Eemeren et al., 2014). The Argument Interchange Format (AIF) (Chesñevar et al., 2006) allows for the analysis of the argumentative structure of a transcript, while Inference Anchoring Theory (IAT) allows this structure be tied back to the illocutionary acts that generated it (Reed & Budzynska, 2011).

In this paper, we present a foundation for inducing dialogue games from analysed transcripts of real inter-human dialogue. We describe the Dialogue Induction Algorithm (*DI-Algorithm*) which extracts sequences of locutions (illocutionary acts and the speakers thereof) from IAT-analysed transcripts and uses those sequences to learn a dialogue grammar. The induced grammar consists of production rules that generate syntactically-valid sequences of dialogue moves, representing the protocol that was followed to produce the original transcript. We illustrate the application of the *DI-Algorithm* by means of a worked example, based on a simple AIF analysis of a mock conversation.

2. BACKGROUND AND MOTIVATION

2.1 Motivation

In real, inter-human dialogues it is rare that an explicit dialogue protocol is followed, and instead it is conventions and norms that dictate the flow of dialogue. Through analysing such dialogues, however, we can expose a protocol encoded within. Speech Act Theory (Searle, 1969) connects propositional reports of dialogue events (e.g. "Bob says it is sunny") via applications of illocutionary force (e.g. "asserting") with their propositional contents (e.g. "it is sunny"). By chaining together the applications of illocutionary force, we can identify an abstract sequence of acts that can subsequently be generalised into a dialogue game representing the encoded protocol (e.g. "asserting" can follow "questioning").

Inducing dialogue games from transcripts of real conversations provides significant benefits both to academia and beyond. As an example of the former, an analysis of a dialogue game induced from transcripts of political debates (e.g. the United States presidential debates) could reveal previously unobserved phenomena that occur in this particular type of discourse.

Beyond academia, application areas range from social computing (where software could support more natural dialogues with mixed-initiative argumentation systems such as Arvina (Lawrence et al., 2012)), to professional training and development (such as successfully inducing a dialogue game that models a courtroom dialogue, providing a training tool for new and existing lawyers).

2.2 Inference Anchoring Theory

Inference Anchoring Theory, IAT (Reed & Budzynska, 2011), provides a set of mechanisms founded in philosophy and linguistics for connecting together two perspectives on argumentation: those that focus on inferential structures and those that focus on inter-agent interactions. IAT unites three pieces of machinery for understanding, representing, manipulating, supporting and creating arguments. First, is a lightweight generalisation of theories of argument macrostructure that identify components of arguments as (more or less) propositions along with relationships between those components focusing on inference and conflict. Second, is an approach to handling discourse structure which focuses on protocol-governed (i.e. rule-governed) transitions between discourse moves. Sitting between these two is the third component, a derivative of speech act theory (Searle, 1969) which connects propositional reports of dialogue events (such as "Bob says it is sunny") via applications of illocutionary force (such as "asserting") with their propositional contents (such as "it is sunny"). The propositional contents are the bread and butter of the first, inferential-oriented, component of IAT; the reports of discourse events are part of the second, interaction-oriented component of IAT; and the third, illocution-oriented component of IAT ties them together. As a result, IAT allows us to explain how an instance of an inference (say from p to q) can be *anchored* in the transition from an agent's challenge of q ("why do you think q is the case?") to the respondent's reply ("well, because p"). Thus IAT captures the intuition that inferences often exist precisely in virtue of the transition from one dialogue move to another – neither challenging q nor asserting p are intrinsically arguing anything. It is only by virtue of asserting p immediately after having q challenged that an inference is established. IAT provides a precise mechanism for showing how inferences are anchored in dialogical behaviour.

2.3 Formal grammars and grammar induction

A *formal grammar* is a set of production rules that allow sentences to be constructed in a formal language, based on two sets of symbols: *terminals*, which cannot be replaced through the application of production rules; and *non-terminals*, which are replaced (by terminals and non-terminals) through applying production rules.

The Chomsky Hierarchy (Chomsky, 1956) is a classification of different types of formal grammar based on the language they generate and the restrictions on the form that rules take. Broadly, there are four levels to the hierarchy, described below where: A and B are non-terminal symbols; α, β and γ are strings of terminal and/or non-terminal symbols; and a is a single terminal symbol.

1. Unrestricted languages (also called recursively enumerable), where production rules have the form $\alpha \rightarrow \beta$; that is, the string α can be replaced by the string β.

2. Context-sensitive languages, where production rules have the form $\alpha A\beta \rightarrow \alpha\gamma\beta$; that is, the non-terminal A can be replaced by the string γ iff A is preceded by the string α and succeeded by the string β.

3. Context-free languages, where production rules have the form $A \rightarrow \gamma$; that is, the non-terminal A can be replaced by the string γ regardless of the context of A.

4. Regular languages, where production rules have the form $A \rightarrow a$ and $A \rightarrow aB$; that is, the non-terminal A can be replaced either by

a single terminal, or a single terminal followed by a single non-terminal.

When written, grammars generally have their rules abbreviated such that those with the same left-hand side have their right-hand sides expressed in a single set. For instance, $A \rightarrow \{\beta, \gamma\}$ represents two context-free rules $A \rightarrow \beta$ and $A \rightarrow \gamma$.

As a simple example to illustrate a formal grammar, consider the following rules which describe valid short phrases in the English language, where "_" is a visible space:

$S \rightarrow \{ARTICLE\_NOUN, S\_VERB\_PREPOSITION\_ARTICLE\_NOUN\}$
$NOUN \rightarrow \{cat, hat, mat\}$
$ARTICLE \rightarrow \{a, the\}$
$VERB \rightarrow \{sat, ran\}$
$PREPOSITION \rightarrow \{in, on\}$

The production rule S describes valid phrases in the language: a sentence can consist of an article and a noun, or (recursively) a sentence followed by a verb, a preposition, an article and a noun. Valid phrases in this language include (but are not limited to):

the_cat
a_mat
$the\_cat\_sat\_in\_the\_hat$
$the\_hat\_sat\_on\_the\_cat$
$a\_mat\_ran\_on\_the\_cat\_sat\_on\_the\_mat$

The last sentence illustrates an important principle of formal grammars — production rules generate syntactically-valid sentences in the language with no consideration for semantics.

Formal grammars can be used for representing dialogue game specifications. In this paper, we will provide examples of context-sensitive grammars that describe dialogue games; one through a worked example, the others being derived from real data.

2.4 Grammar induction

Grammar induction, sometimes referred to as grammatical inference, is the process of inducing a grammar based on a set of sentences (a language) that grammar can generate. Duda et al. (2001) provide a simple algorithm for grammar induction. This algorithm takes a set of training sentences and returns a grammar that can generate those sentences. Since it is possible for two or more grammars to generate the same language, such algorithms are sometimes expanded to accept as additional input a set of “negative” sentences that are known not to be derivable in the grammar. This limits the potential production rules and thus increases the likelihood of yielding a unique grammar.

In broad terms, given a set of positive (training) examples $\mathcal{D}^+$ and a set of negative examples $\mathcal{D}^-$, the algorithm takes each sentence in $\mathcal{D}^+$ in turn and adds to the grammar the required production rule(s) that allow that sentence to be generated, provided said rule(s) do not allow a sentence in $\mathcal{D}^-$ to be generated.

This algorithm also imposes two constraints: 1) that the alphabet (terminal symbols) of the resultant grammar be only those used in the training sentences; and 2) that every production rule in the grammar is necessary in order to regenerate the training sentences.

In our approach to dialogue grammar induction we specify an algorithm that is bound by these constraints, but does not necessarily yield a only a single grammar and thus does not require a set of negative examples. We do not impose a uniqueness requirement because it is possible for some sequences of locutions to be generated by two or more different dialogue protocols, which in turn are described by different dialogue grammars. Our intention is to unearth all possible dialogue grammars from a given training set of sequences.

3. THE *DI-ALGORITHM* FOR DIALOGUE GRAMMAR INDUCTION

In this section, we present the *DI-Algorithm* for inducing a dialogue grammar from transcripts of real conversations. The algorithm takes as input AIF-IAT analyses of transcripts and generates the production rules for a context-free grammar that represents a simplified version of the dialogue protocol.

In common with other grammars, those induced by the *DI-Algorithm* consist of both terminal and non-terminal symbols. The terminals are locutions extracted from the analysed transcripts; the non-terminals are generated by the induction process when creating the production rules.

The *DI-Algorithm* breaks down into a 4-stage process:

1. Locution sequence extraction — from an AIF analysis, extract sequences of locutions in the order in which they were uttered.

2. Minimal valid sequence identification — for each extracted sequence, identify minimal non-atomic sequence(s) that are valid with respect to the transitions in the original sequence.

3. Centre enrichment— find (sub-)sequences in the original sequence that are expansions of the minimal input (i.e. sequences that enrich the minimal expansion with further locutions)
4. Rule generalisation — modifying rules to reduce the size of the grammar by replacing sub-sequences in right-hand sides with the left-hand sides of rules whose right-hand side is that sub-sequence

We now describe each stage of the process in detail with a running example to illustrate each concept. We specify sub-algorithms as logic programs for the sake of brevity.

3.1 Locution sequence extraction

The first stage of inducing a dialogue grammar is extracting sequences of locutions from an IAT analysis. Consider the following mock conversation between Alice and Bob, as they discuss what activity they should do:

> **Alice:** *We should go to the cinema.*
> **Bob:** *Why do you say that?*
> **Alice:** *Because we enjoy watching films.*
> **Bob:** *We should go to the park.*
> **Alice:** *Why do you say that?*
> **Bob:** *Because we enjoy the outdoors.*
> **Alice:** *OK, we should go to the park.*

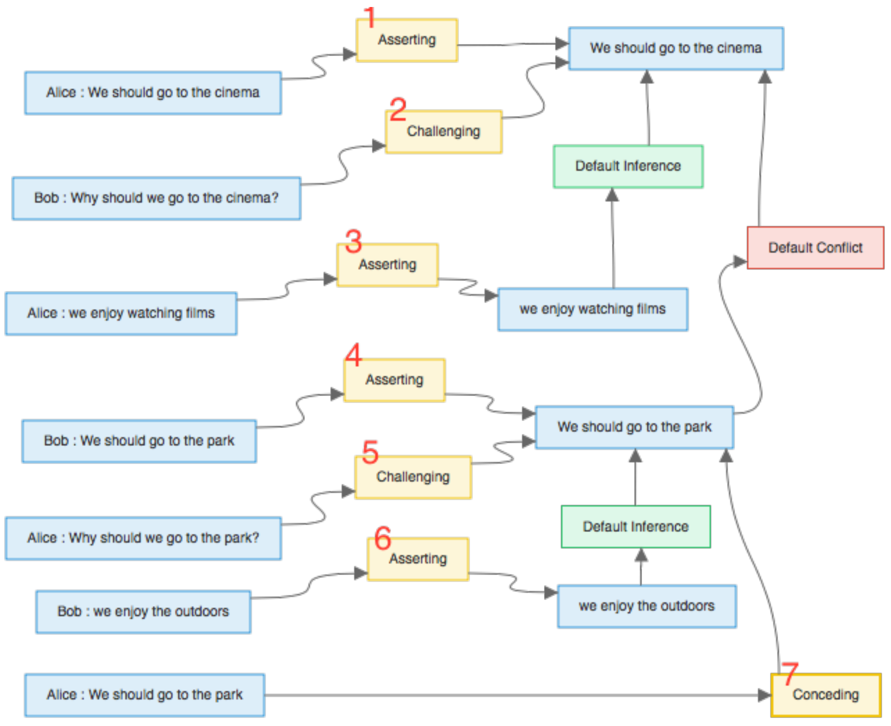
An analysis of this conversation is shown in Figure 1, with the numbers representing a simplified representation of AIF timestamps (for clarity, we omit the full IAT analysis and anchorings which are not relevant for our purposes). From this analysis, we use the AIF timestamps to extract the following ordered list, S_1, of locutions identified by the analyst, where the subscript denotes the speaker:

$$\mathcal{S}_1 = (assert_A, challenge_B, assert_A, assert_B, challenge_A, assert_B, concede_A)$$

This sequence represents a valid dialogue in the protocol whose grammar we are attempting to induce. To fully illustrate the algorithm, we will use two other valid sequences, derived from other analyses, in a set $\mathcal{S}$ of three thus:

$$\mathcal{S} =$$

$$\left\{ \begin{array}{l} \mathcal{S}_1\colon (assert_A, challenge_B, assert_A, assert_B, challenge_A, assert_B, concede_A), \\ \mathcal{S}_2\colon (assert_A, challenge_B, assert_A, challenge_B, assert_A, concede_B), \\ \mathcal{S}_3\colon (assert_A, assert_B, assert_A, concede_B) \end{array} \right\}$$



1
Asserting
We should go to the cinema
Alice : We should go to the cinema
2
Challenging
Default Inference
Bob : Why should we go to the cinema?
3
Asserting
Default Conflict
Alice : we enjoy watching films
we enjoy watching films
4
Asserting
We should go to the park
Bob : We should go to the park
5
Challenging
Default Inference
Alice : Why should we go to the park?
6
Asserting
Bob : we enjoy the outdoors
we enjoy the outdoors
7
Alice : We should go to the park
Conceding

Figure 1 – Example analysis

3.2 Minimal valid sequence recognition

The second stage of the algorithm is to identify minimal valid sequences of locutions, based on the sequences extracted from the AIF analyses. Given a sequence $\mathcal{S}_i$, the minimal valid sequence $\mathcal{S}_{i_\perp}$ is the shortest subsequence of $\mathcal{S}_i$ that is non-atomic (i.e. longer than one element) such that the first and last locutions in $\mathcal{S}_{i_\perp}$ are, respectively, the first and last locutions in $\mathcal{S}_i$. The non-atomic condition prevents minimal sequences of only one locution; in the simplest case, this can arise from a sequence $\mathcal{S}_j = (assert, challenge, assert)$. Without the non-atomic condition, $\mathcal{S}_{j_\perp} = (assert)$ because the first and last locutions are the same.

We look for minimal valid sequences in order to identify the shortest dialogues permitted by the protocol. This is based on the assumption that, for a given sequence $\mathcal{S}_i = (s_1, \ldots, s_n)$, s_1 is a valid starting locution in the protocol and s_n is a valid terminating locution. Identifying the shortest permitted dialogues provides the basis for the rules of the dialogue grammar.

Minimal valid sequences are established by either starting at the first locution and working forward through the sequence to find the closest occurrence of the last locution (front-to-back), or starting at the final locution and working back through the sequence to find the closest occurrence of the type and speaker of the first locution (back-to-front).

Once again using S_1 as an example, we first label each locution in the sequence to provide clarity:

$l_1 : assert_A$

$l_2 : challenge_B$

$l_3 : assert_A$

$l_4 : assert_B$

$l_5 : challenge_A$

$l_6 : assert_B$

$l_7 : concede_A$

The first and last locutions are, respectively, $assert_A$ and $concede_A$. The closet occurrence of $assert_A$ to the end of the sequence is locution l_3. Thus in this example the minimal valid sequence is that bounded by l_3 and l_7 inclusive:

$$\mathcal{S}_{1\perp} = (assert_A, assert_B, challenge_A, assert_B, concede_A)$$

Algorithm 1 shows the sub-algorithm used to determine the minimal valid sequence in a given list using the back-to-front approach.

Algorithm 1 Minimal Valid Sequence determination

```
% mvs(+Sequence,-MinimalValidSequence)
mvs([X | Y], [X | Y]):-
        \+ member(X,Y).
mvs([X | T], [X | Y]):-
        next_sublist(X, T, S),
        mvs([X | S], [X | Y]).


mvs([X | Y], [X | Y]):-
        \+ member(X,Y).
mvs([X | T], [X | Y]):-
        next_sublist(X, T, S),
        mvs([X | S], [X | Y]).

% next_sublist(+Head,+ListStartingWithHead,-ListWithHeadRemoved)
next_sublist(X, [X | S], S).
next_sublist(X, [_ | T], S) :-
        next_sublist(X, T, S).
```

The minimal valid sequences for $\mathcal{S}_2$ and $\mathcal{S}_3$ in the training set are:

$\mathcal{S}_{2\perp} = (assert_A, concede_B)$ $\mathcal{S}_{3\perp} = (assert_A, concede_B)$

Using the minimal valid sequences, we arrive at the following initial production rules in the induced grammar, where "D" is a non-terminal start symbol:

$$\mathcal{P} = \left\{ \begin{array}{l} r_1 : D \rightarrow (assert_A, assert_B, challenge_A, assert_B, concede_A), \\ r_2 : D \rightarrow (assert_A, concede_B) \end{array} \right\}$$

3.3 Centre enrichment

The third stage of the algorithm is to identify ways in which adjacent pairs of locutions can be enriched by inserting sequences between them to create longer valid sequences. When a pair in a minimal valid sequence is enriched, a longer valid sequence of locutions is created, representing a longer dialogue. It is these enrichments that generate the production rules for the dialogue grammar.

Generating production rules based on enrichments is a two-stage process; for every enrichment:

1. generate a rule in which the left-hand side is a new non-terminal and the right-hand side is the (un-enriched) pair
2. for each enrichment of a pair, generate a rule in which the left-hand side is the same non-terminal and the right-hand side is the enriched pair

The first step in centre enrichment is to extract each adjacent pair of locutions from the input sequence. Returning to our example training set, the adjacent pairs in the sequence

$\mathcal{S}_1$: $(assert_A, challenge_B, assert_A, assert_B, challenge_A, assert_B, concede_A)$

are:

$p_1 : (assert_A, challenge_B)$ $p_4 : (assert_B, challenge_A)$

$p_2 : (challenge_B, assert_A)$ $p_5 : (challenge_A, assert_B)$

$p_3 : (assert_A, assert_B)$ $p_6 : (assert_B, concede_A)$

Algorithm 2 shows the sub-algorithm for extracting pairs.
Taking each pair in turn sub-sequences of S_1 are identified which start and end with, respectively, the first and second locution in the pair. Algorithm 3 shows the sub-algorithm for centre enrichment.

Algorithm 2 Pair extraction

```
% pairs(+Sequence,-[Pairs])
pairs([],[]).
pairs([_],[]).

pairs([H, T | S], [[H, T] | Z]) :-
pairs([T | S], Z).
```

Algorithm 3 Centre enrichment

```
% remove_last(+Sequence,-SequenceWithLastElementRemoved)
remove_last([_], []).
remove_last([X | Y], [X | T]) :-
        remove_last(Y, T).

% enrich(+Pair,+Sequence,-EnrichedPair)
enrich([], [], []).
enrich([X, Y], [X | T], S) :-
        remove_last([X | T], L),
        \+ last(Y, L),
        enrich([X,Y], L, S).
enrich([X , Y], [_ | T], S) :-
        enrich([X , Y], T, S).
enrich([X , Y], [X | T], [X | T]) :-
        last(T, Y).
```

In the case of S_1, there is only one pair with valid enrichments, p_3, with two in total. These lead to the production rules:

$r_3 : N_1 \rightarrow (assert_A, assert_B)$
$r_4 : N_1 \rightarrow (assert_A, challenge_B, assert_A, assert_B)$
$r_5 : N_1 \rightarrow (assert_A, challenge_B, assert_A, assert_B, challenge_A, assert_B)$

Applying the same process to sequences S_2 and S_3 yields the following production rules:

$r_6 : N_2 \rightarrow (assert_A, challenge_B)$
$r_7 : N_2 \rightarrow (assert_A, challenge_B, assert_A, challenge_B)$
$r_8 : N_3 \rightarrow (assert_A, concede_B)$
$r_9 : N_3 \rightarrow (assert_A, challenge_B, assert_A, challenge_B, assert_A, concede_B)$
$r_{10} : N_3 \rightarrow (assert_A, assert_B, assert_A, concede_B)$

3.4 Rule generalisation

In their current form, the rules can generate only trivial dialogues, because the right-hand side of each rule consists only of terminals. The final stage of the algorithm is to therefore generalise the rules such that, where possible, sub-sequences of terminals are replaced by non-terminals, allowing rules to grow and thereby accurately describe the dialogue protocol.

A rule is generalised if a sub-sequence in its right-hand side can be replaced by the left-hand side of another rule (i.e. a non-terminal). If there is a production rule r_1 whose right-hand side is a sub-sequence in another rule r_2's right-hand side, replace the sub-sequence in r_2 with the left-hand side of r_1. For instance, the right-hand side of rule $r_3 : N_1 \rightarrow (\textbf{assert}_\textbf{A}, \textbf{assert}_\textbf{B})$ is a sub-sequence of the right-hand side of rule $r_{10} : N_3 \rightarrow (\textbf{assert}_\textbf{A}, \textbf{assert}_\textbf{B}, assert_A, concede_B)$.

Thus our final set of production rules, including the start rules, is:

$$\mathcal{P} = \left\{ \begin{array}{l} r_1 : D \rightarrow (N_1, challenge_A, assert_B, concede_A), \\ r_2 : D \rightarrow (N_3), \\ r_3 : N_1 \rightarrow (assert_A, assert_B), \\ r_4 : N_1 \rightarrow (N_2, N_1), \\ r_5 : N_1 \rightarrow (N_2, N_1, challenge_A, assert_B), \\ r_6 : N_2 \rightarrow (assert_A, challenge_B), \\ r_7 : N_2 \rightarrow (N_2, N_2), \\ r_8 : N_3 \rightarrow (assert_A, concede_B), \\ r_9 : N_3 \rightarrow (N_2, N_3), \\ r_{10} : N_3 \rightarrow (N_1, N_3) \end{array} \right\}$$

Following rule generalisation, it is possible to filter the set of rules in order to eliminate those that can produce the same sequences as other rules, but are less general. This, however, is a step we don't yet wish to implement because we expect propositional content of locutions to have an impact on the filtration. Since it is our intention to build on the *DI-Algorithm* to account for propositional content (see Section 5), we do not want to introduce any processes that will require significant modification.

4. RELATED WORK

Here, we briefly compare and contrast our approach to dialogue game induction with existing work on both dialogue protocol/game induction, and grammar induction in general.

Similar grammar induction techniques are used in (Alexandersson & Reithinger, 1997) and (Geertzen, 2009) to extract dialogue structures from corpus data. In both cases, however, the authors examined the problem in the context of dialogue act prediction — given a grammar induced from analysed or marked-up transcripts, how accurately can that grammar predict the next act in a dialogue using the same protocol. This differs from the present work which uses dialogue grammar induction as a step towards inducing an accurate dialogue protocol. It was further noted in (Geertzen, 2009) that algorithms for inducing dialogue grammars could be tested against data that has been annotated by dialogue games. We believe that our work develops a solid foundation for this by using recent developments in automated dialogue game execution (the Dialogue Game Execution Platform (Lawrence et al., 2012)) to generate annotated data against which our induced grammar is tested.

Grammar induction in general is a significant problem in computational linguistics and natural language processing. Approaches to grammar induction are split broadly into the classic machine learning approaches of supervised and unsupervised induction. In supervised induction, the algorithm is provided with sets of annotated sentences from which it learns sentence structure; in unsupervised induction, the algorithm must learn the structure from scratch (Clark & Lappin, 2010).

A variety of different techniques have been used in unsupervised grammar induction. In recent years, much work has built on the generative model of (Klein & Manning, 2002). This model relies on part-of-speech (POS) annotations, where models are trained using sequences of POS tags instead of raw tokens of text. An alternative approach, that parses raw text, is the *common cover links* (CCL) parser (Seginer, 2007). This approach, however, is hard to extend (Ponvert et al., 2011).

Our approach to dialogue grammar induction is set apart from unsupervised natural language grammar induction by the nature of the input to the *DI-Algorithm*. In unsupervised natural language grammar induction, the aim is to induce a grammar from a set of raw, unannotated sentences. Dialogue grammar induction, on the other hand, uses analysed transcripts where the illocutionary acts and speakers thereof are explicitly identified thus removing the need to first identify the type of act an utterance represents; in other words, dialogue grammar induction has already identified the non-terminal symbols ($assert$, $challenge$ etc.), with the terminal symbols being the specific locutions complete with propositional content. In (unsupervised) natural language grammar induction, it is first necessary to identify those non-terminal symbols (noun, verb, adjective etc.) based only on the words themselves in the text.

Our approach does, however, have similarities with supervised natural grammar induction that relies on part-of-speech annotations. These POS annotations are similar to the identified illocutionary acts in the transcripts that feed the *DI-Algorithm*. A key difference is that a dialogue has clearly defined start and end rules that are used in the determination of minimal valid sequences that underpin the *DI-Algorithm* and make dialogue grammars more restrictive than natural language grammars.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a foundation for inducing dialogue games from analysed transcripts of real, inter-human dialogues, using techniques from formal grammar induction. We described and tested the *DI-Algorithm* for dialogue game induction. Using dialogical AIF, incorporating Inference Anchoring Theory, the *DI-Algorithm* extracts sequences of locutions (illocutionary acts and the speakers thereof) and uses them as a training set for learning a formal grammar. The grammar is a simplified representation of the dialogue protocol, whereby the production rules can verify sequences of locutions that are valid with respect to the original protocol.

In future work, we will design and implement and framework for testing the accuracy of the *DI-Algorithm*. This framework will attempt to induce a dialogue grammar using data generated from executing existing dialogue protocols, with the aim being to arrive at a grammars that accurately reflect those original protocols. Although the test framework is not yet specified, we envisage that accuracy will be measured based on false positives and false negatives in the induced grammars.

Further future work will be to further develop the *DI-Algorithm* so it can induce grammars not just from sequences of locutions, but also their propositional content. Achieving this will allow a DGEP-executable DGDL specification to be generated and, by building on our test framework proposed above, we can compare this new specification to the original one used to generate the test data, thus adding an extra step of verification.

What we have done in this paper is provide a foundation for inducing dialogue games from analysed corpora of real conversations.

ACKNOWLEDGEMENTS: This work was supported by the Leverhulme Trust under grant number RPG-2013-076. The authors are also grateful to colleagues in the Centre for Argument Technology for their comments and feedback on earlier versions of this work.

Alexandersson, J., & Reithinger, N. (1997). Learning dialogue structures from a corpus. In *Proceedings of the Fifth European Conference on Speech Communication and Technology, EUROSPEECH 1997*.

Black, E., & Hunter, A. (2009). An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems*, *19*, 173–209.

Chesñevar, C., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., & Willmott, S. (2006). Towards an argument interchange format. *The Knowledge Engineering Review*, *21(4)*, 293–316.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*, 113–124.

Clark, A., & Lappin, S. (2010). Unsupervised learning and grammar induction. In Clark, A., Fox, C., & Lappin, S. (Eds.), *The Handbook of Computational Linguistics and Natural Language Processing*, pp. 197–220. Wiley-Blackwell.

Duda, R., Hart, P., & Stork, D. (2001). *Pattern Classification*. Wiley-Interscience.

Geertzen, J. (2009). Dialogue act prediction using stochastic context-free grammar induction. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, pp. 7–15.

Hamblin, C. (1970). *Fallacies*. The Chaucher Press.

Klein, D., & Manning, C. D. (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 128–135. Association for Computational Linguistics.

Lawrence, J., Bex, F., & Reed, C. (2012). Dialogues on the Argument Web: Mixed initiative argumentation with Arvina. In Verheij, B., Szeider, S., & Woltran, S. (Eds.), *Proceedings of the Fourth International Conference on Computational Models of Argument (COMMA 2012)*, pp. 513–514, Vienna, Austria. IOS Press.

McBurney, P., & Parsons, S. (2002a). Dialogue games in multi-agent systems. *Informal Logic*, *22(3)*, 257–274.

McBurney, P., & Parsons, S. (2002b). Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information*, *11*, 315–334.

Ponvert, E., Baldridge, J., & Erk, K. (2011). Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics - Volume 1*, HLT ’11, pp. 1077–1086, Stroudsburg, PA, USA. Association for Computational Linguistics.

Reed, C. (1998). Dialogue frames in agent communication. In Demazeau, Y. (Ed.), *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS 1998)*, pp. 246–253, Paris, France. IEEE Press.

Reed, C., & Budzynska, K. (2011). How dialogues create arguments. In van Eemeren, F. H., Garssen, B., Godden, D., & Mitchell, G. (Eds.), *Proceedings of the 7th Conference on Argumentation of the International Society for the Study of Argumentation*.

Searle, J. (1969). *Speech Acts*. Cambridge University Press.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Vol. 45, p. 384. Association for Computational Linguistics.

Snaith, M., Lawrence, J., & Reed, C. (2010). Mixed initiative argument in public deliberation. In De Cindo, F., Macintosh, A., & Peraboni, C. (Eds.), *Proceedings of the fourth international conference on Online Deliberation (OD2010)*, Leeds, UK.

Van Eemeren, F. H., Garssen, B., Krabbe, E. C., Henkemans, A. F. S., Verheij, B., & Wagemans, J. H. (2014). *Handbook of argumentation theory*. Springer Berlin.

Walton, D. N. (1984). *Logical Dialogue-Games and Fallacies*. University Press of America.

Walton, D., & Krabbe, E. (1995). *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, New York.