Automated Parsing of Dialogue Games in Open Ended Dialogue

Author Matthew Marchesani

Adviser Nick Webb

June 13, 2014

Abstract

In Natural Language Processing, the ability to model the structure of language is a key to solving problems. We currently can label words an utterances according to their meaning and communicative function. My research involves creating a structure out of these lower components by identifying patterns in dialogues. I will approach this problem by analyzing a corpus of open-ended dialogue. My research will contribute important information to Dialogue Management Systems' about language structure that can be implemented in its architecture and will help enable human-like generation and understanding of dialogue.

1 Introduction

Research has shown, through studies of naturally occurring dialogue, that people seem to interact according to pattern conventions [9]. Dialogue relies on conventional knowledge of what response each participant can expect from the other. This can be one framework from which coherent dialogues are built. Figure 1 is a diagram of a Dialogue Game (DG).

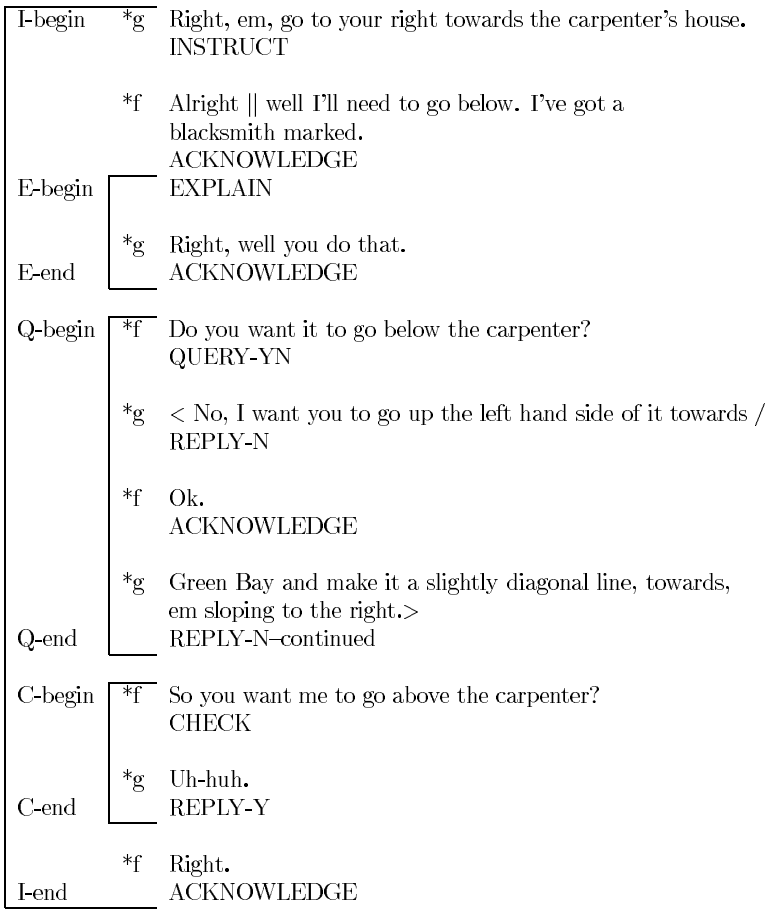

Figure 1: Dialogue Game Example

Field et al [5] state that DGs are exchanges between speakers that fulfill a goal. Additionally, Conversational Theory states that these patterns form the structure of any and every possible dialogue [6] [3]. Modeling dialogue structure is an inherently fundamental step toward a larger overall goal of having Dialogue Management Systems (DMS) that are able to understand and generate spontaneous dialogue [12].

The nature of task-based dialogue results in a goal oriented view of language [1]. Open-ended dialogue is not always goal oriented, but some goal oriented dialogue may be embedded in non-goal oriented dialogue. We believe that many of the DGs found in task based dialogue will be found in the open-ended dialogue. Also I believe that there will be phenomena in the dialogue that are not captured with the tags. And so I will try to tag all of the new phenomena that I find. Additionally, I will be looking for utterances that do not serve a structural function and which can be omitted from the corpus. Omitting these utterances is similar to taking the stop words out of sentences. The essence of the sentence is not lost, and when the stop word like utterances are removed, the structure of the dialogue is not lost. After that, I plan to use machine learning to gain statistical information about DGs in the corpus. I am particularly interested in DGs that are involved in canonical forms of DGs and embedded forms of DGs. I believe that gaining probabilities of games being embedded or appearing canonically will make computationally understanding spontaneous dialogue much easier. DG starting points and ending point in open-ended dialogue are more complex than they are in the task based dialogue [2]. The data I will use is from the Switchboard corpus. I am analyzing an annotated text version rather than an audio version. This simplifies the research problem to one that is approachable for the amount of time I have to spend.

Current state of the art, human-agent DMS, does not have the robustness of dialogue that humans do. Even more interesting, agent-agent systems currently exhibit significant incapabilities due to distinctly different implementations of dialogue structure components. [11]. The solution to my research problem will contribute to DMSs' architecture and help produce DMSs with human-like understanding and generation of dialogue between human-agent and agent-agent systems. DMS that can understanding and generating conversations effectively as humans can pick up the slack of voice recognition modules, as well. When a human-agent systems is not able to recognize spoken input, it will be able to use its knowledge of dialogue structure to make a human-like guess at what the person said [4].

2 Related Work

I have not been able to find much research on DG tagging in open-ended dialogue. And I have found some research on tagging task-based dialogue, and there is a fair amount of theoretical work on dialogue structure, including DGs. The current state of task based DG knowledge is divided among a handful of theoretical approaches [7]. Attempts to support several approaches with empirical data have failed. These failure were due to the fact that the intention based approach and the psychological commitment based approach for modeling dialogue structure considers the psychological states of the participants. Predictions about psychological states of participants cannot be verified and are difficult to produce reliably. These failure have resulted in a movement in the direction of shallow discourse analysis (i.e. no consideration of psychological states of participants) [10] [11]. The most popular being the social commitment based dialogue structure analysis.

2.1 Dialogue Structure

Conversational Theory states that there are dialogue structures and they are conventions of natural language [9]. Researchers have empirically shown that dialogues are built up from structures that are formed from sequences of utterances that contribute individual functions of the dialogue [3]. The sequences of communicative functions create coherence in conversations [9] [11] [3].

In the past, complex dialogue behavior has been emulated with network and stack architectures. Implementing a stack provides for a way to reenter a state of a dialogue that was interrupted [5]. For example, the current DG will be on top of the stack, and the nested is pushed on the stack before the previous top was popped off. Then when the new current is popped off, the previous is continued. [8]

2.2 Dialogue Acts

Currently, many researchers use of Dialogue Act (DA) sequences to describe dialogue structure [12] [13]. A DA is a label for an utterance's communicative function [12], and include labels such as, question, statement, and backchannel. Their general purpose is to serve as initiations or responses. DAs can be automatically labeled [12] [5] [13]. This research provides a significant foundation for my research.

There are DA sets that are completely domain-independent, the popular most widely used being the Dialogue Act Markup in Several Layers (DAMSL) tag set [1] [12]. This set is used by several computational frameworks that, accurately and reliably, automatically tag utterances with labels [13]. This set of DAs may be able to infer DGs that are the building blocks for conversational game structure, which reflects the goal structure of the dialogue [2]. My research will be using this set.

2.3 Dialogue Games

Communicative Theory researchers state that dialogue has structure, and that the structure is a fundamental concept of communication [9] [1] [11] [3]. A coherent conversation happens due to constraints that dialogue games enforce on a dialogue [9] [1] [11] [3]. And through using dialogue games any dialogue can be synthesized by the abstract structures. The social-commitment based approach explains the structure in a data driven way [11] [3]. It says that the interlocutor binds their self to prepositions or actions, within the dialogue [11] [3]. The dialogue game approach to the structure of dialogue can be social-commitment based and therefore be built upon easily verifiable data.

DGs capture shared conventional knowledge that people have about communication, and are patterns in dialogue made up of one or more turns of a participant [9]. And each turn is made of a DA. In an attempt to be able to model the structure of dialogue, Conversational Theory has begun to look at theses patterns closely and given them the name DG [9].

In a DG coding scheme it is important to identify the purpose of the game and give a description of how games are related to each other [1]. At any point in a dialogue a participant can refuse continue the initiated game. This can be in the form of ignoring the initiation and initiating a move of their choice. It is also possible for a explicit refusal. For example, rebuffing a question with the statement "Let's not talk about this."

All games hinge on the fact that, in general, questions are followed by answers, greetings by greetings, statements by acceptances or denials, ect [9] [1] [3]. And because of this nature game analysis has a fundamental approach of dividing into two groups. One being initiations, which set up some expectation about what will follow. The other is response, and it fulfills the expectation that is set up [1]. Games can also be further divided by purpose, either getting information or providing information [9].

3 Methods and Design

The Switchboard corpus, which contains 1155 5-minute telephone conversation that make up a total of 205,000 utterances, was tagged with a shallow discourse tagset. The tags indicate the dialogue acts of the utterances. The tagging of the corpus was part of a project call Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation. I will use this tagged corpus of open-ended dialogue to. I will use the DA with which that the Switchboard corpus was labeled already. This is not only convenient but the tags have been formally evaluated for reliability. I will use machine learning software to extract statistical data about DGs from the Swtichboard corpus.

4 Approach

4.1 Corpus Preparation

The corpus is in XML format. There are elements for dialogues, turns and utterances. Turns have attributes for their number in the dialogue and their speaker. Utterances have attributes for their number in the turn and type of DA. The words said by the speaker are in CDATA elements that are contain inside of utterances.

I spent a significant amount of effort trying to get the corpus in a easily processable form. It turns out that this process resulted in the awareness that the work I planned to do required more than one version of the corpus. A main feature of the different corpora is that initially the corpora contained actual XML elements for turns. At the end of the research, I rarely used this format. Corpora with dialogues that contained only utterances with speaker attributes to indicate who took the turn were mostly used.

First I prepared the corpora by I replaced short tags DA with full length ones. The full length tags gave me the ability to read data and write code containing the tags much easier than translating short tags to their meanings or memorizing all of them.

The original version of the corpus that I used did not have uniformly sequenced turns. There were adjacent turn by the same speakers. These turns contained only one utterance. These turns did not serve any function and made traversing the corpus difficult. And so I reformated the corpus dialogue by dialogue. When repeating turns by one speaker occurred, I inserted the utterances of following turns at the end of the initial turn. This continued up until a turn change to the initial turn, I deleted the turns that followed the initial turn. This essentially is conflation of adjacent turns made by one speaker. Making the turn sequence uniform like that meant that there would be no turns by the same speaker next to each other.Furthermore, when a person has to read the corpus with many turns containing single utterances, they can become tired and frustrated quickly. Later on this would clearly make the corpus traversal algorithms much easier to write than ones that would be made for a corpus with many turns containing only one utterance.

After conflating the turns, within each one, there could be more than one utterance. This feature corresponded to the multiple turns by the same speaker in the initial corpus. Because there were more than one utterance in a turn, there could be adjacent utterances of the same type. This is like they way that more than one turn by the same speak were found adjacent to one another in the original corpus. These utterances were conflated the way that the turns were. Although when conflating utterances of the same type, the result is not applicable to the general analysis. For example, if two wh-questions by the same speaker get conflated, a very interesting feature of the corpus could be lost. This is a reason why there were multiple corpra made. This version with conflated utterances is useful and must be analyzed. But it is not perfect, and it only contributes a part of the research that is done. Even though, conflation of utterances needs to be done with careful consideration, this version of the corpus yielded valuable insights that are

discussed later in this paper.

The utterances that were conflated then had multiple !CDATA elements in them. These could be conflated. The conflation of these resulted in large blocks of text that were really easy to read. Of course, because these CDATA elements were childeren of utterances that were conflated there was a trickle down effect of the possiblity of combining thing that should not be combined. As stated above, all this conflation is only part of the preparation of a single version of the corpus it is not a final version of the corpus.

In all the versions, I renumber the dialogues, turns and utterances. The numbering happened to the children of elements. For example, if there were n turns in one dialogue and a turn had n utterances, they would be correspondingly labeled.

I also removed attributes that were there previously and had no relevance to my work. I learned that, in general, to many XML attributes in elements tends to make difficult to read XML.

As mentioned above with the conflation of utterances, the need for having more than one version of the corpus came about over the course of several weeks. Another key influence in this decision was that at first there were algorithms that did all types of additional work. For example, an algorithm would travers the corpus and conflate the utterances before it would do pattern counting. A conflated corpus would make the algorithm take much less time. If an algorithm has to do conflation while it traverses, for example, that algorithm will be much slower than one that processes a corpus which has already had conflation done to it. So time savings in the long run showed up as well. Furthermore, it was much easier to make different versions of corpora, and then write an algorithm that works for many or all of them. Versus, writing many complicated algorithms that work on one main version of the corpus.

At times, I needed to delete elements from the corpus, like when I removed the turns. I learned that iterating backwards over the collections, going from n - 1 to 0, made it possible to remove elements while iterating. Iterating from 0 to n - 1 and removing as you go create a way for elements to be completely skipped over.

On interesting thing that I did to prepare the corpora for the DG exploration and parsing, included combining statement-non-opinion DAs and statement-opinion into one statement type DA. Here are some statistics for the combination of the two statement DAs in to only one:

Statement-Non-Opinion	75403
Statement-Opinion	26529
Statement (sum of two previous)	101932

Another interesting thing that I did to prepare the corpra is, I removed 7 types of utterances. I did this because these utterance types did not control the flow of the dialogue in a way that I would consider during my research. I made a version of the corpus with utterances with the following DA tags removed from turns. This resulted in some turns with no utterances in them (totaling 3% of the corpus). These tags are taken out since their influence on turn changes in dialogues are not easily pinned down. For my research the influence on turn changes that these DAs have are not concrete enough or readily interpretable enough. Here is a run down:

Non-Verbal	3678
Other	1222
Uniterpretable	15682
Quotation	980
Self-Talk	103
Acknowledge-Backchannel	38372
Backchannel-in-Question-Form	1057
Grand Total	61094

I will run the scripts on the version of the corpus that I just created. So I removed the empty turn and conflated turns, then utterances, then CDATA, where appropriate. During this process, I looked at the corresponding text in the dialogues where turns ended up with no utterances. I clearly saw that removing the turns did not impact the flow of the dialogue in any noticeable way. In fact, the dialouges seemed to flow smoothly over the parts were gaps by turns containing zero utterances were formed joined up by removing those turns. Removing the utterances causes 6% of the turns to have zero utterances and 10% of the utterances to be removed.

The original did not have utterances removed or statement types combined or utterances conflation. Utterance conflation will not change the number of turns in a dialogue. These are three main formats of the corpus:

	Original	Without Utterance Conflation	With Utterance Conflation
Turns	116187	50829	50829
Utterances	204915	143821	71675

Below are comparisons of counts of DAs in the corpora without utterance conflation and with utterance conflation:

	Without Utterance Conflation	With Utterance Conflation
Statement	101932	35184
Agree/Accept	11157	8349
Appreciation	4797	4461
Yes-No-Question	4727	4499
Yes-Answers	3042	2716
Conventional-Closing	2585	1821
Wh-Question	1989	1926
No-Answers	1383	1186
Response-Acknowledgement	1309	1228
Declarative-Yes-No-Question	1252	1214
Hedge	1227	1192
Summarize/Reformulate	961	912
Affirmative-Non-Yes-Answers	849	807
Action-Directive	752	662
Collaborative-Completion	723	705
Repeat-Phrase	698	690
Open-Question	657	645
Rhetorical-Questions	578	524
Hold-Before-Answer/Agreement	557	510
Reject	345	291
Signal-Non-Understanding	299	299
Negative-Non-No-Answers	298	292
Other-Answers	286	276
Conventional-Opening	225	178
Or-Clause	210	208
Dispreferred-Answers	207	199
3rd-Party-Talk	117	71
Offers-Options-Commits	110	105
Maybe/Accept Part	105	99
Downplayer	104	89
Tag-Question	98	98
Declarative-Wh-Question	85	84
Apology	79	79
Thanking	78	76
Grand Total	143821	71675

Here are the questions taken from the data above. This is a reminderthat conflation can remove features of the dialogues. When questions are conflated, structures in the dialogues can be removed. The questions in the dialogues are the focus of my research, and that is why they are highlighted:

	Without Utterance Conflation	With Utterance Conflation
Yes-No-Question	4727	4499
Wh-Question	1989	1926
Declarative-Yes-No-Question	1252	1214
Open-Question	657	645
Rhetorical-Questions	578	524
Tag-Question	98	98
Declarative-Wh-Question	85	84
Grand Total	9386	8990

One reason to do modifications to the corpus is that it is a simplification tool. The research problem can be solved by simplifying the problem and then building up sophistication. The solution to the simplified problem may potentially be added to so that it solves the more complete goal. And solving the simplified problem is an easier starting point than approaching the overall problem with a direct approach.

I learned from the preprocessing of the corpus is that it's ideal to think about this stuff early as possible. Then data does not need to be regathered, and avoiding dealing with incorrect or incompelte initial data. Regathering a ton of data in light of a useful modification that has just been thought of can be quite time consuming.

For example, a little late in the game, I replaced sd and sv with a DA called statement. This is going to change my patterns. This replacement should have been done early. It is a simplification of what exists in the corpus. An sd or an sv would not change the flow of the dialogue any differently, and so replacing both of them with a single statement DA would should not lose any critical feature of the dialogue. By examining the patterns with separate sd and sv DAs, a more fine grained analysis is done. And that is not what I intend to do during this round of analysis.

After hand annotating, I determined that I needed to change the corpus to one without turns. Automating the corpus with turns would have created a difficult to read structure.

4.2 Tools Intro

Since the formate is XML, there are many different programming languages that I could have used to analyze the corpus. I chose Python 2.7 mainly because how easy it is to use Natural Language Toolkit as a module. Furthermore, Python is a language that is quick and easy to write algorithms in. There are a few options for module to work with the XML Dom in Python. I used the Python Mini Dom. During my preliminary research about XML, several of reliable sources on the web championed Mini Dom. And I easily found blogs and websites, including Stackoverflow, containing more tech information about Mini Dom than the other available options for XML in Python. The Mini Dom gave all of the functionality that XML specification requires. The Tools That I Used While Processing the Corpus. NLTK POS tagger uses pen tree bank pos tags. I easily wrote the scripts that traverse the corpus. Making them customized to do the work that I

wanted to do to the corpus involved simply adding logic to extract turns that sandwich changes in speakers and utterances with certain DA types.

4.3 Initial Work

I can make the dictionary as the script comes across keys. I started by making the dictionary with all possible ways a pattern could happen. But many of those ways did not happen. And so there were zero values in the dictionary. Now that I make the dictionary keys as I go, I do not need to desparsify the dictionary. There are no keys that correspond to a value of zero. When I output the data, I do it in with frequencies mapping to patterns, like a histogram. But the Python dictionary cannot be sorted. So I turn the dictionary in to a list and invert the key values. Then I map the frequencies to patterns and sort the list in descending order of frequencies.

This is exploratory work and uses problem and data simplification often. I also used compositions of what I found in order to get to the next step in the research. First I needed to find out if patterns existed in the dialogues. And so, initially, I did a simplified version of a solution. I did not start a the DGs. I took the stance that DG had to be built up from atomic pieces. In the beginning of my research, I began looking at the atomic level. One goal was to come across patterns of DAs that could be used directly as DG parsing rules.

Since turns in the dialogue are sequenced uniformly from speaker Aat to speaker B, I am going to look at pairs of DAs that sandwich turn changes. Turn changes can happen A to B and B to A. I will make a n by n hash of the DAs. I will increment the frequencies of the DA pair value when it is found sandwiching a turn change. I used these counts to compare with the frequencies of DA pairs that occur irrespective of any turn change. This data could be used to make inferences, like, if a DA pair happens only at a turn change, the initial DA signals a turn change. It is important to keep in mind that, the DAs in these patterns maybe the first of several utterances a turn or the last of several utterances in a turn. Furthermore, they could be the only utterances in a turn. And so, one must be familiar with all of the different lenses that the data is looked at under.

Now that I have two main versions of the corpus that have turn elements or do not have turn elements, I need a traversal algorithm for each of these main versions. I traversed the corpus by getting collections of element types and indexing the elements in the collection, rather than using sibling traversal. There are text elements between siblings and this has to be accounted for in the latter method. I am conflating the utterances in the patterns. The conflation algorithm looks at the adjacent utterance (before or after the utterance its adjacent to, depending on where in the pattern the algorithm is) and treats the adjacent and initial as one if the DA labels are the same. Being careful not to skip any utterances while traversing is extremely important. It is easy for an iteration to be off by one if there is a boundary case that is written sloppily. This is after removal and conflation. If an entire turn is only one type of utterance, than the turn was skipped. In this, if either has only one type, then next turn is brought up. But I want to keep this target turn and check both turns from this perspective. If there is only one type of utterance in turn i, but there is more than one type in turn i+1, I conflate where I can and keep the pattern. Before, I incremented i. Not incrementing results in something like DA turn DA DA or DA DA turn DA, because one side is conflated.

The first analysis went through every dialogue, every turn, and every utterance. It took counts of the patterns of adjacent utterances regardless of what speaker made them. For example, if there were two turns, A and B with two utterances each $A_1A_2B_1B_2$, three patterns would be counted A_1A_2, A_2B_1, B_1B_2. Following this I went through each dialogue and each turn For example, if there were two turns, A and B with two utterances each $A_1A_2B_1B_2$, one pattern would be counted A_2B_1. from turn 0 to turn n-1 and

take the ith turn and the ith+1 turn an incremented a counter, in the form of a Python dictionary that had patterns mapped to counts. There were 35^2 possible permutations and X actually occurred. Initially, I used two for loops, one nested in the other, to create the dictionary with initial counts equal to zero. Then I realized that as I went through the dialogues, I could add the patterns to the dictionary as they happened. This saved quite a bit of computation. And it also saved having to do the work of checking for patterns that had zero counts later on. Sometimes when I printed out the counts I would not want the zero count patterns to be included, and by using this alternative method, I did not need to include checking for zeros in my printing algorithm, which worked out well for me. No sparsity. Reverse to make a histogram. Cannot sort a dictionary, so turn it in to a list.

I did this with three and four utterances in the following fashion. For three utterances, the for loop would iterate from i = 0 to i = (the number of utterances - 1) - 2. And in the body of the loop indexing the second and third positions in the pattern could be done with i+1 and i+2 without going out of bounds when being at the end of the dialogue. For four utterance patterns, the loop would end at the (number of utterances - 1) - 3. At the end there would be no indexing out-of-bounds from the body of the loop.

I found the frequency of the pattern any where in the dialogue and the frequency of the pattern if there were two turns, A and B with two utterances each $A_1A_2B_1B_2, A_1A_2B_1$ and $A_2B_1B_2$. For three utterances, I found the frequency of the pattern anywhere in the dialogue and at turn changes, A and B with two utterances each $A_1A_2B_1B_2, A_1A_2B_1$ and $A_2B_1B_2$. For four utterances, I found the frequency of the pattern anywhere in the dialogue and at turn changes, A and B with two utterances each $A_1A_2B_1B_2, A_1A_2B_1B_2$. By using these two sets of counts I could determine which patterns never occur at turn changes, always occur at turn changes, or what the frequency of a pattern at a turn change when it occurs both inside of turns and at the changing of turns.

As I added to the pattern, the percentages at turn changes grew higher. That was because less of the longer patterns happened in the middle of turns, then sandwiching a turn change. This is easy to imagine being the case.

This is the older way of doing utterance conflation as the iteration over the corpus happens. But if I need to keep track of where conflation happens then I need to use this approach. I could have added attributes to utterances that tell the amount of conflation that happened. Although, I never tried this approach. This is after removal and conflation. If an entire turn is only one type of utterance, than the turn was skipped. Although, I want to also look at when both, only one of the turns, has only one type of utterance. In this, if either has only one type, then next turn is brought up. But I want to keep this target turn and check both turns from this perspective. s s ynq ya s s s would not be the same as s s s ynq ya s. There is a Python file for doing this. The corpus that the files us must have turns. And obviously not have utterances conflated. The format of the results is like: 15, statement-non-opinion[2], statement-opinion[1], agree/accept, statement-non-opinion. The bracketed number tells how much conflation happened.

A great transition to parsing from initial work is that I decided to parse patterns that happen frequently. I looked at the top 10 possible games being the patterns that occurred most frequently at turn changes. I decided to go with question answer, opening opening, and closing closing. I created DG paring rules for them by using the and DAs turn change sequence that correspond to the patterns.

4.4 Question Answer(QA) DGs

When doing simple parsing of QA DGs, the main characteristic is that if the utterances in the answering speakers turn are all part of the answer unless the utterances are questions. Everything up to the first question, or the turn change, is considered the answer. a more sophisticated parser could look at ynq ya na

Right now, I am using mini DGs. This is a strategy that parses nested games from the inside out. (I saw simple two level DGs in the corpus.

I initially started by parsing DG rules with specific DAs. These games were based on DA patterns that I found which had high frequencies. (EG yq ya na) I decided to simplify my approach. I ended up taking anything following a question that is not a question to be an answer. For example ().

any question followed by a statement is a dg

This is simple parsing because it only looks at question answer game, and it considers everything forllowing a question up to the next turn change to be the answer. This does not parse nested q/a games (eg q q a a would get parsed as q DG a)-question answer is a pattern, but question question answer answer is not being parsed at this point the conflation would have given an idea of whether or not nearly all the corpus would be parsed or not.

The conflation will cause sibling statements to be treated as one, and so more of the corpus would be involved in a game than what actually was involved in a game. The dialogues should have been looked at and not the corpus. There may be dialogues that are completely parsed even if the corpus is only partially parsed.

I have two parsers one for with turns and one for without turns. This is because initially I used the corpus with turn elements, and later on it became apparent that when parsing the DGs it was difficult to annotate where the DG started and where it ended when using the turn element version of the corpus. A bit of planning for how an algorithm would traverse a corpus with no turn elements but keep track of turn changes happened.

I found out that I definitely needed speaker attributes and not turn elements for speakers during the following process. Using speaker attributes has a few advantages over speaker turn elements. One is that the parsed DG corpus can later be run on my pattern frequency counting algorithm and the patterns that are not involved in a DG can be counted. I parsed the corpus and still used the statistical algorithms for the non-parsed corpus. Introducing DG elements would break those algorithms. I considered replacing turns that create DGs with DG elements. But because turns can have more than one utterance, some turns are involved in a DG at the beginning of the turn and some are involved in a DG at the end of the turn, or both. And removing a turn that has only one end involved in a DG will not capture what happens at the other end. And it is important for what is happening at the other end to remain in the corpus for analysis later on. For example, if turn i and turn i+1 form a DG, but i+1 and i+2 do not form a DG, and I replace i and i+1 with a DG, the non pattern between i+1 and i+2 will no longer be in the corpus. I decided to to insert a DG in between turns that formed a pattern. Doing this would make a pattern that the algorithm initially used to parse the game no longer found by the algorithm. The parsing algorithm fails to see the pattern because a DG has been inserted between the two turns and it splits the pattern. This is advantageous because I can run the algorithm or a fairly similar one on a corpus that has been parsed and old patterns will not be parsed a second time. Also, the parsed DG corpus can later be run on my pattern frequency counting algorithm and the patterns that are not involved in a DG can be counted.

At first, I thought that the percentage of questions in a dialogue could be an indicator for a dialogue that would have a large percentage of its utterances parsed.

Dialogue	Percentage Is Questions
34	35%
893	34%
7	33%
17	32%
967	30%
261	29%
198	28%
56	28%
809	28%
891	27%

	Without Utterance Conflation
Yes-No-Question	3893
Wh-Question	1531
Declarative-Yes-No-Question	1024
Open-Question	498
Rhetorical-Questions	212
Tag-Question	82
Declarative-Wh-Question	49
Total Parsed	7289
Total Utts in Games	31821
Percentage of Corpus	22%

Without Utterance Conflation

Dialogue	DGs	Utterances in DGs	Total Utterances	Percent
38	4	44	59	75%
1150	14	86	120	72%
888	12	56	78	72%
766	4	45	63	71%
129	3	54	81	67 %

	With Utterance Conflation
Yes-No-Question	3881
Wh-Question	1522
Declarative-Yes-No-Question	1023
Open-Question	493
Rhetorical-Questions	212
Tag-Question	82
Declarative-Wh-Question	48
Total Parsed	7261
Total Utts in Games	19308
Percentage of Corpus	27%

With Utterance Conflation

Dialogue	DGs	Utterances in DGs	Total Utterances	Percent
1	3	13	18	72 %
100	15	42	62	71%
371	16	44	62	71%
888	12	32	48	67%
7	6	22	33	67%

First, I put DG elements at the turn change. Second, I added DG start and DG end attributes to utterances. Adding attributes to utterances with out turns made the parsed corpus much easier to read. The values of the DG start and DG end attributes were the corresponding counts for the DGs in a dialogue.

4.5 Noun Similarity QA DG Parsing

With this parser the breadth of the answer is now being limited to the utterances that contain the same topic as the question or beginning of the answer (ie the immediately following utterance). To be able to parse the corpus in this way, I came up with the idea that I would use a special version of the corpus. I made a version of the corpus with noun attributes added to utterances. Then when I want to look up the nouns for an utterance all that I have to do is grab the value for the the corresponding noun attribute.

To make the nouns attribute. I removed all special markup in the CDATA text (eg <laughter >, {F uh, }). And I made sure that the algorithm did not remove apostrophes to preserve the words in the utterances. Because the text in the CDATA elements usually is not perfectly grammatic, parsing this data to find the nouns was not as reliable as I would have liked. Any thing that will work to prevent not parsing important topic nouns needed to be maximized. Then I part of speech tagged words in utterances with NLTK's part of speech (POS) tagger. Since the utterances in the dialogues did not have perfect grammatical structure, the POS tagging did not work with perfect accuracy (need to find a way to improve this. After the text was POS tagged, I took all of the nouns with the following part of speech tags out

Tag	Type	Function
NN	Noun	singular or mass
NNS	Noun	plural
NNP	Proper noun	singular
NNPS	Proper noun	plural
PRP	Persona	pronoun
PRP$	Possessive	pronoun

This list of nouns included what are considered to be stopwords. So I took the stopwords out of the list. I added an attribute called nouns to the utterances in the corpus, and the list of nouns that I extracted from the utterances got placed in the value of the utterances' noun attributes. Since each utterance subsequently contained a noun attribute with all its nouns, comparing the subject of an utterance to the subject of another utterance could conveniently take place. I modified an existing parser in a way that took the subject of an utterance into account when searching for the end of a QA DG.

The parser used the following algorithm: when there is a change in turns, if the first speaker gives a question and the second speaker does not respond with a question parsing will take place. The nouns from the first speaker's question and the second speaker's response are considered the QA nouns. If there are no QA nouns the game start and end are simply marked appropriately in these two adjacent utterances. If this is not the case, and the utterance following the second speaker's first utterance is another utterance by the second speaker, and it's not a question, the utterance's nouns are compared to the QA nouns. If the sense is within .7 of each other, they are included in the game. This continues until the next utterance is a speaker one, or a speaker two question, or a speaker two non-question that does not contain any noun with the topic of any QA nouns.

This is an example taken from the corpus of a game:

Speaker A:
DA = wh-question, nouns = night:
And what night is it on?
Speaker B:
DA = hold-before-answer/agreement, nouns=[]
Well, we watched it,
DA = statement, nouns = times excuse friday night
The times we've seen it, it's been like on Satur-, or, excuse me, Friday night.
DA = hedge, nouns = couple times
We've only seen it a couple of times.

I took some statistical information on parsed QA DGs from the corpus parsed with noun similarity. In the corpus with no utterance conflation, there are 161 games that have 7 or more utterances. In the corpus with no utterance conflation, 1 game has 5 or more utterances. In the corpus with utterance conflation, these results sense because many of the utterances that are included in a game in the corpus with no utterance conflation are likely would get conflated and that is what makes the numbers go down.

	Without Utterance Conflation
Yes-No-Question	3893
Wh-Question	1531
Declarative-Yes-No-Question	1024
Open-Question	498
Rhetorical-Questions	212
Tag-Question	82
Declarative-Wh-Question	49
Total Parsed	6383
Total Utts in Games	15728
Percentage of Corpus	11%

Without Utterance Conflation

Dialogue	DGs	Utterances in DGs	Total Utterances	Percent
33	4	25	47	53%
891	6	23	49	47%
623	9	32	71	45%
34	5	23	58	4%
370	19	46	121	38%

	With Utterance Conflation
Yes-No-Question	3881
Wh-Question	1522
Declarative-Yes-No-Question	1023
Open-Question	493
Rhetorical-Questions	212
Tag-Question	82
Declarative-Wh-Question	48
Total Parsed	6456
Total Utts in Games	14487
Percentage of Corpus	20%

With Utterance Conflation

Dialogue	DGs	Utterances in DGs	Total Utterances	Percent
1075	10	24	39	62%
33	4	10	17	59%
955	11	26	49	53%
772	6	16	31	52%
18	9	20	39	51%

4.6 Nested Games

The following statistics are for the simple QA parser. In the corpus without utterance conflation, there are 9386 questions. And the number of games parsed is 7289. So there are 2097 questions that are not parsed, which is 0.22 percent of the corpus. In the corpus with utterance conflation, there are 8990 questions and 1729 parsed games. That mean 0.19 percent, or one third, of the questions are potentially part of a nested game. I pulled text for dialogues that had questions that were not parsed. This showed many of the the non-parsed games being part of nested structures, like expected.

The following statistics are for the noun similarity QA parser. In the corpus without utterance conflation, there are 9386 questions. And the number of games parsed is 6383. So there are 3003 questions that are not parsed, which is 0.32 percent of the corpus. In the corpus with utterance conflation, there are 8990 questions and 2534 parsed games. That mean 0.28 percent, or one third, of the questions are potentially part of a nested game. I pulled text for dialogues that had questions that were not parsed. This showed many of the the non-parsed games being part of nested structures, like expected.

5 Future Work

Finding the topic of chunks of the dialogues would be a way to help improve the noun parsing that is done to add noun attributes to utterances. The incorrect grammar of some utterances makes it important nouns left out of the noun attributes. By dividing the dialogues up by topic, then the set of topic nouns can be searched for in utterances. Therefore, a simple look up can be done at the same time that the POS tagging is. The improvement over the current implementation would definitely be worth the work to format the corpus in this way. Although, I have not figured out a way to represent the sections of topic in the XML. This would have to be perused as well. It is different from giving utterances noun attributes because there will be groups of utterances that are part of a topic. This additional meta data would need to be located somewhere. Looking at the topic of the utterance that follows the DG ending can help build up sections of dialogue corresponding to topics. This level of analysis could lead to interesting statistic on how many DGs per topic happen in dialogues. Once the topic of a DG can be established, then the average number of DG per topic can be found. This is valuable because it can be used to find probabilities of DGs continuing on a certain topic after this number.

Since only a third of the corpus is parsed in to DGs, it would be interesting to find what is in between the games. The first step is looking at how many utterances how many turns. Next, the types of DAs would need to be looked at. While looking at the things in between the DGs, there would definitely be some nest DGs found.

The current parsers are essentially parsing the inner most part of nested games. So moving out a level of nesting is a great thing to work toward in the future. Finding the questions that tend to be at the beginning of nested games would be valuable at this step as well. The statistics would help when they are used by a dialogue management system to do probabilistic reasoning.

Of course, finding new type of games is also a good topic for future work. A new type of game maybe built up from my simple QA DGs. For instance, a QA DG followed by an acknowledge/agree accept DA could be considered a QA with Acknowledgment DG. Additionally, patterns with questions followed by or-clauses are fairly common and neither of my parsers do anything with them. This functionality is a good investment which likely will lead to other ideas for future work.

References

[1] Jean Carletta, Amy Isard, and Jacqueline Kowtko. *HCRC dialogue structure coding manual.* HCRC Publications, University of Edinburgh, 1996.

[2] Jean Carletta, Stephen Isard, Gwyneth Doherty-Sneddon, Amy Isard, Jacqueline C Kowtko, and Anne H Anderson. The reliability of a dialogue structure coding scheme. *Computational linguistics*, 23(1):13–31, 1997.

[3] Guillaume Dubuisson Duplessis, Nathalie Chaignaud, Jean-Philippe Kotowicz, Alexandre Pauchet, and Jean-Pierre Pécuchet. Empirical specification of dialogue games for an interactive agent. In *Advances on Practical Applications of Agents and Multi-Agent Systems*, pages 49–60. Springer, 2013.

[4] Ulrich Endriss, Wenjin Lue, Nicolas Maudet, and Kostas Stathis. Competent agents and customising protocols. In A. Omicini, P. Petta, and J. Pitt, editors, *Proceedings of the 4th International Workshop Engineering Societies in the Agent World (ESAW-2003)*, volume 3071 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 168–181. Springer-Verlag, 2004.

[5] Debora Field, Simon Worgan, Nick Webb, Mark Hepple, and Yorick Wilks. Automatic induction of dialogue structure from the companions dialogue corpus. In *Proc. of Fourth International Workshop on Human-Computer Conversation*. Citeseer, 2008.

[6] Joris Hulstijn. *Dialogue models for inquiry and transaction*. University of Twente, 2000.

[7] Joris Hulstijn and Nicolas Maudet. Uptake and joint action. *Cognitive Systems Research*, 7(2):175–191, 2006.

[8] Antonis Kakas, Nicolas Maudet, and Pavlos Moraitis. Flexible agent strategies and societal communication protocols. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the 3rd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2004)*, pages 1434–1435. ACM Press, July 2004. Extended abstract.

[9] James A Levin and James A Moore. Dialogue-games: Metacommunication structures for natural language interaction. *Cognitive science*, 1(4):395–420, 1977.

[10] Ian Lewin and Mill Lane. A formal model of conversational game theory. In *Procs. of the 4th workshop on the semantics and pragmatics of dialogue (Gotalog 2000)*. Citeseer, 2000.

[11] Nicolas Maudet and Brahim Chaib-Draa. Commitment-based and dialogue-game-based protocols: new trends in agent communication languages. *The Knowledge Engineering Review*, 17(2):157–179, 2002.

[12] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.

[13] Nick Webb and Michael Ferguson. Automatic extraction of cue phrases for cross-corpus dialogue act classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1310–1317. Association for Computational Linguistics, 2010.