The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

# Building Dialogue Structure from Discourse Tree of a Question

**Boris Galitsky**
Oracle Corp. Redwood Shores CA USA
boris.galitsky@oracle.com

**Abstract**

We propose a reasoning-based approach to a dialogue management for a customer support chat bot. To build a dialogue scenario, we analyze the discourse tree (DT) of an initial query of a customer support dialogue that is frequently complex and multi-sentence. We then enforce what we call *complementarity* relation between DT of the initial query and that of the answers, requests and responses. The chat bot finds answers, which are not only relevant by topic but also suitable for a given step of a conversation and match the question by style, argumentation patterns, communication means, experience level and other domain-independent attributes. We evaluate a performance of proposed algorithm in car repair domain and observe a 5 to 10% improvement for single and three-step dialogues respectively, in comparison with baseline approaches to dialogue management.

## Introduction

Answering questions, a chat bot needs to reason to properly select answers from candidates. In industrial applications of search, reasoning is often substituted by learning from conversational logs or user choices. It helps to make search more relevant as long as a similar question has been asked many times. If there is no data on previous similar question, which is frequently the case, a chat bot needs to apply some form of reasoning to select from candidate answers (Wilks 1999).

Most frequent type of reasoning is associated with topical relevance, it requires an ontology and is domain-specific. Difficulties in building domain ontologies are well known, and in this study we are take a different reasoning-based approach. Once a set of candidate answers or replies are available, how to select most suitable ones? The suitability criteria are two-dimensional: 1) topical relevance; and 2) an appropriateness not associated with topic but instead connected with communicative discourse. Whereas topical relevance has been thoroughly investigated, chat bot's capability to maintain the cohesive flow, style and merits of conversation is an underexplored area.

When a question is detailed and includes multiple sentences, there are certain expectations concerning the style of an answer. Although a topical agreement between questions and answers have been extensively addressed, a correspondence in style and suitability for the given step of a dialogue between questions and answers has not been thoroughly explored. In this study we focus on assessment of cohesiveness of question/answer (Q/A) flow, which is important for a chat bots supporting longer conversation. When an answer is in a style disagreement with a question, a user can find this answer inappropriate even when a topical relevance is high. Matching rhetoric structures of questions and answers is a systematic way to implement high-level reasoning for dialogue management, to be explored in this work.

A problem in communicative discourse occurs mostly for complex questions (Chali et al 2009, Galitsky 2017), arising in miscommunication, a lack of understanding, and requiring clarification, argumentation and other means to bring the answer's author point across. Rhetoric disagreement is associated with a broken dialogue and is usually evident via the means an answer is communicated, explained or backed up.

Let us start with an example of a customer support dialogue, where an agent tries to figure out a root cause of a problem (Fig.1). The system finds candidate answers with the keywords and phrases from the initial query, such as *Google Earth, cannot see, attention* and others. Which candidate answers would be the best to match the communicative discourse of the query?

We define a customer support dialogue as a sequence $Q, A_1, C_1, A_2, C_2, \ldots,$ where $Q$ is an initial query describing a problem, $A_1$ is an initial recommendation and also a clarification request, $C_1$ is a response to this request, $A_2$ is a consecutive recommendation and clarification request, $C_2$ is a response to $A_2$, and so forth. Figure 4 shows our model structure for a typical customer support dialogue. Our goal is to simulate a broad spectrum of dialogue structures via correspondence of discourse trees of utterances. This way

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

once $Q$ is given, the chat bot can maintain the list of candidates for $A_i$.

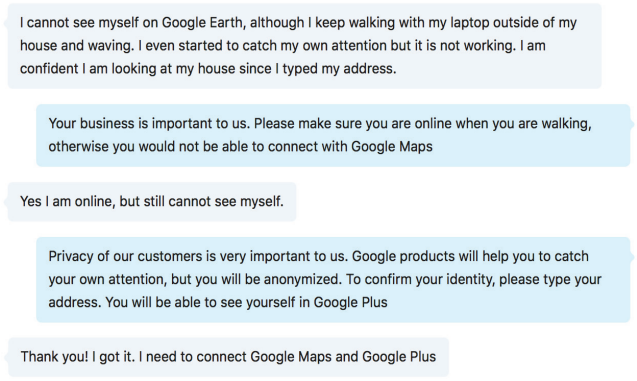


*Fig. 1: Example of a customer support dialogue.*

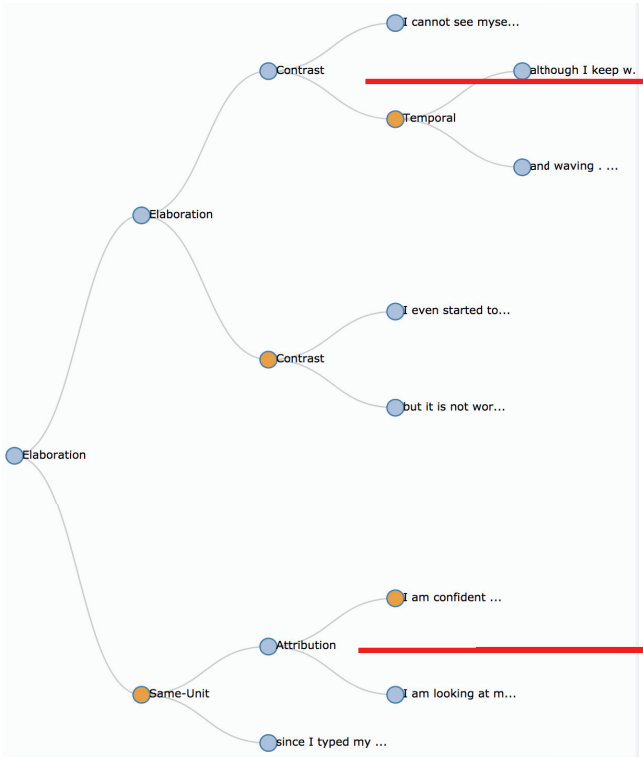


*Figure 2: Discourse tree of a question. (Joty et al 2013) visualization is used.*

## Maintaining Communicative Discourse for Q and A

To maintain communicative discourse, we employ the notion of discourse tree (DT). Rhetoric Structure Theory (RST, Mann and Thompson 1988) models logical organization of text, a structure employed by a writer, relying on relations between parts of text. RST simulates text coherence by forming a hierarchical, connected structure of texts by means of DTs. Rhetoric relations are split into the classes of coordinate and subordinate; these relations hold across two or more text spans and therefore implement coherence. These text spans are called elementary discourse units (EDUs).

DTs for the $Q$, and DT for the sequence of two answers are shown in Figs. 2 and 3 respectively. The chain of RST relations with entities are *Elaboration [see myself Google Earth]-Contrast [walk laptop house ]-Temporal [waiving]* on the top of DT-Q is addressed by the chain *Elaboration [online]-Same_Unit [walking]-Elaboration [not able connect*] in the first answer $A_1$. The second answer $A_2$ attempts to addresses in a complete way the issues raised in the second part of the Q (expressed by sequence *Elaboration [confident] -Same_Unit [looking at my house]-Attribution [typed address])* by two sequences *Elaboration [privacy]-Elaboration [anonymized] - Contrast[catch attention]* and *Elaboration [privacy]-Elaboration [confirm identity] – Enablement [type address]*.

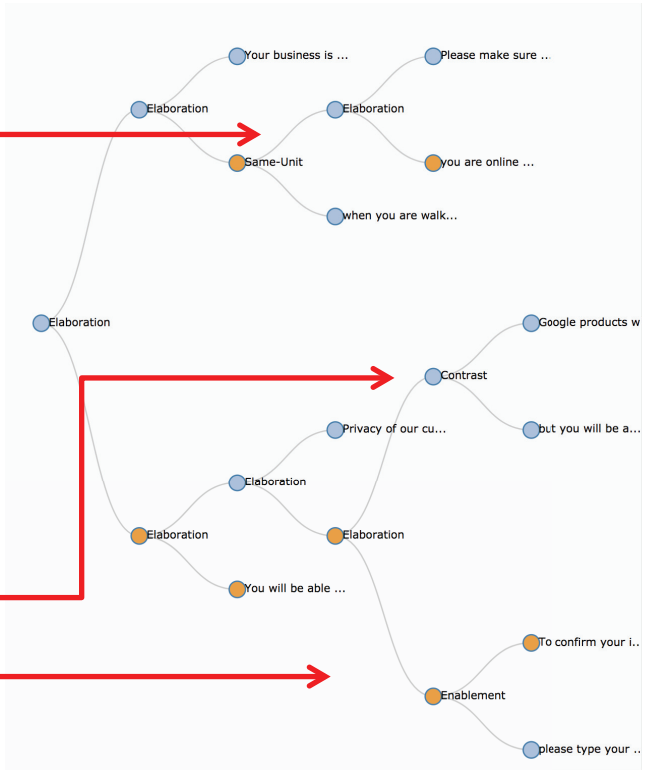


*Figure 3: Discourse tree of a sequence (pair) of answers.*

We use the format

*RST-relation [phrase]* for each node of a DT. The *phrase* part is an OR query including all linguistic phrases of a given EDU (shown here in an abbreviated form).

The main observation here is that the question itself gives us a hint on a possible sequence of answers, or on the order the issues in the question are raised. One can look at the DT-Q and form a dialogue scenario (*first do this, obtain confirmation, then do that ...*). Since a dialogue is built

from available answer fragments (e.g. from conversational logs), we take candidate answers, form candidate DTs from them and see if they match DT-Q.

A simple rule would be for a chain of RST relations for an *A* to be a sub-chain of that of a *Q*. But this rule turns out to be too restrictive and even invalid in some cases. Our observation is that *DT–A* does not have to copy *DT-Q* or its parts, but instead rely on certain complementary relations between *DT–A* and *DT-Q*:

*complement(DT–Q, DT-A).*

Search algorithm that supports the dialogue is as follows:

1) Build *DT-Q*;
2) Split *DT-Q* into parts $Q_1$, $Q_2$,... to correspond to $A_1$, $A_2$,...;
3) Form search query for $A_1$ from $Q_1$ in the form *RST-relation [phrase]* ;
4) Run the search against the set of dialogue utterances and obtain the list of candidate answers for the first step $A_1$ *candidate*;
5) Build $DT\text{-}A_{1\text{candidate}}$ for each candidate and approve/reject each based on *complement(DT–Q, DT-*$A_{1\text{candidate}}$*)*;
6) Respond to the user with the selected $A_1$ and receive $C_1$;
7) Form search query for $A_2$ from $Q_1$&$C_1$;
8) Repeat steps 4) and 5) for $A_2$ , respond to the user with the selected $A_2$ and receive $C_2$;
9) Conclude the session or switch to a human agent

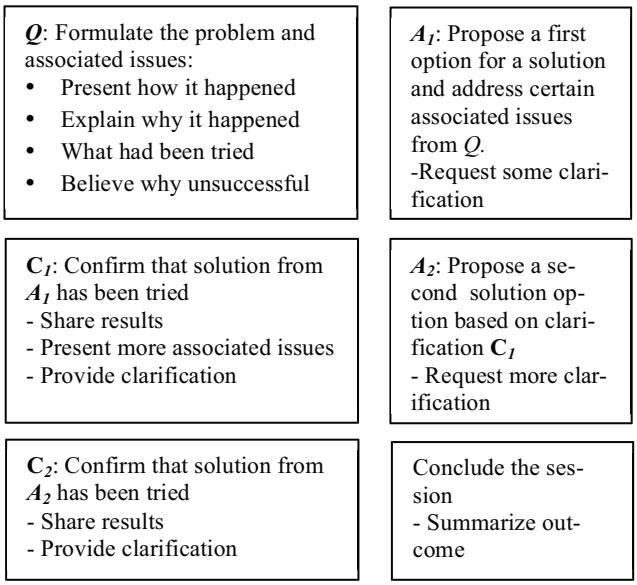


*Figure 4: The model of a customer support dialogue*

Hence the dialogue management problem can be formulated as a search with constraints on DTs and can be implemented via traditional search engineering means plus discourse parsing, when an adequate set of chat logs is available. Discourse-tree based dialogue management does not cover all possibilities of assuring smooth dialogue flows but provides a plausible mechanism to select suitable utterances from the available set. It allows avoiding solving NL generation problem for dialogues that is a source of a substantial distortion of conversation flow and a noise in meaning of utterances.

Some issues related to dialogue management do not need to be achieved via DTs. For example, an implementation of *clarification* feature can be hard-coded and does not require specific rhetoric relations. When a user asks a broad question, the chat bot forms topics for this user to choose from. Once such a topic is selected, the full answer is provided (Fig. 5).

Hence the developed chat bot relies on a computational measure for how logical, rhetoric structure of each utterance is in agreement with a previous utterance. The system forms a DT representation for a $Q_i/A_i$ pair based on RST and applies machine learned *complement* relations (which is a focus of the following section). We will solve a $Q_i/A_i$ pair classification problem of relating them into a class of valid (correct, appropriate answer) or invalid pairs.

## Learning *complement* relation

Given a multi-sentence question, we try to machine-learn from a well-tuned search engine, which answers are good with respect to everything other than topic (to form a positive dataset), and which answers are bad (to form a negative dataset). We define complementary relation as the one that holds between a given question and a good answer, and does not hold between this question and a bad answer. Hence we can machine-learn *complement* relation for a set of Q/A pairs.

To form the above training set, one needs to employ a search engine that has a different criteria (from the current study) on which parts of answer is good and which are not good. For example, it can be answer popularity, or search rank, which is learned by search engine on the basis of the high number of searches for the same query and user selection.

To accumulate Q/A pairs tagged as good and bad, we run a high number of arbitrary queries against short texts. Since we need longer queries to assure the match is non-trivial, we take a Yahoo! Answers dataset (Webscope 2017) and formed the web search queries from the first sentences of the questions for Q/A pairs. We rely on Microsoft Cognitive Services (Bing Search engine API) to run these web searches. Then we select those search results that are short texts (4-6 sentences) suitable for parsing and discourse analysis. We then take *DT-Q/DT-A* pairs as elements of the training set.

The answers from the top 10+ pages of search result form our positive dataset. It includes the fragments of texts

considered by the search engine to be of high relevance. For the negative dataset, we take the fragments with matched keywords from the set of lower ranked (100-1000+) search results pages. The classifier now needs to differentiate between positive *DT-Q/DT-A* pairs (from the top search results) and negative *DT-Q/DT-A* pairs (from the lower ranked search results).

Since both *Q* and *A* determines if *DT-A* is the best for *Q* or not, the objects in our learning setting is neither *Q* nor *A* but instead Q/A pairs. In real time search it means that a number of candidate pairs will be scored by the *complement* relation, having high or low score. Alternatively, following along the lines of search engineering approaches, we do all learning offline, prepare a lookup of classes of *DTs* for *Q* and find optimal respective classes of *DTs* for *As*.

A search engine dealing with complex Qs such as Yahoo! Answers, for example, needs a systematic approach to assess the *complement* relation and select the most suitable answers among the relevant ones. DTs' features could be represented in a numerical space where a classification into valid or invalid Q/A pairs would be conducted; however, structural information on DTs would not be leveraged.

Conversely, the *complement* relation can be assessed in terms of maximal common sub-DTs, but it is computationally intensive and too sensitive to errors in DT construction. Therefore a DT-kernel learning approach is selected which applies SVM learning to a set of all sub-DTs of the DT for Q/A pair. Tree kernel family of approaches is not very sensitive to errors in parsing (syntactic and rhetoric) because erroneous sub-trees are mostly random and will unlikely be common among different elements of the training set.

Given a positive dataset for the *complement* relation and a negative dataset for it, we attempt to recognize if a given Q/A pair is covered by *complement* relation. Notice that a DT for Q and a DT for A can be arbitrary, but only DTs for a pair can be linked by the *complement* relation. Then this algorithm is applied to perform passage re-ranking of answers to achieve the highest possible *complement* relation maintaining relevance.

Tree Kernel (TK) learning for strings, parse trees and parse thickets is a well-established research area nowadays. The parse tree kernel counts the number of common sub-trees as the discourse similarity measure between two DTs. TK relies on the operation of generalization '^' which is applied at the level of parse and discourse trees, phrases, and words (Galitsky et al 2012). A version of TK has been defined for DT by (Joty and Moschitti 2014). (Wang et al 2013) used the special form of TK for discourse relation recognition. In this study we extend the TK definition for the CDT, augmenting DT kernel by the information on CAs.

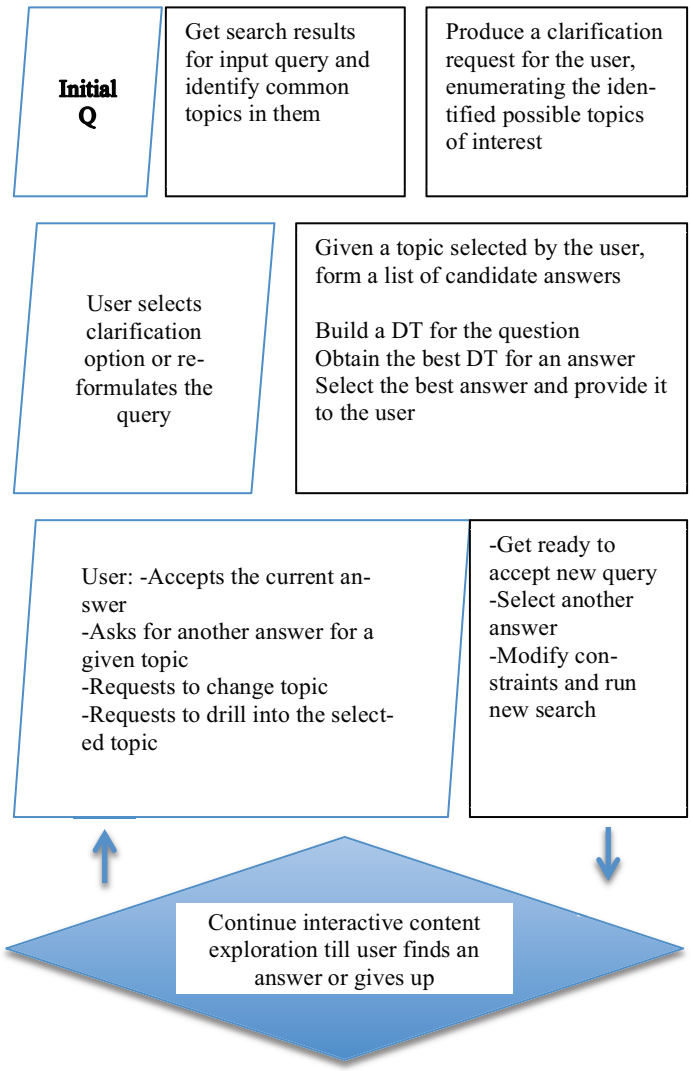


*Figure 5: Clarification structure of a customer support dialogue*

A DT can be represented by a vector V of integer counts of each sub-tree type (without taking into account its ancestors):
$V(T)$ = (# 1,…, # of sub-trees of type i, # of sub-trees of type n). Given two tree segments $DT_1$ and $DT_2$ , the tree kernel function is defined: $CDT_1 \wedge CDT_2 =$
$K(DT_1, DT_2) = <V(DT_1), V(DT_2)> =$
$\Sigma_i V(DT_1)[i], V(DT_2)[i] = \Sigma_{n1}\Sigma_{n2}\Sigma_i \; I_i(n_1) * I_i(n_2)$, where $T$ is a tree, $n_1 \in N_1$ , $n_2 \in N_2$ where $N_1$ and $N_2$ are the sets of all nodes in $DT_1$ and $DT_2$ , respectively; $I_i(n)$ is the indicator function: $I_i(n)$ = {1 iff a sub-tree of type $i$ occurs with root at node; *0* otherwise}.

As an alternative to TK family of approaches, we used a direct DT similarity comparison by maximal common sub-DT (Galitsky et al 2013). The higher the cardinality of this sub-tree, the higher is the similarity score. The hypothesis

here is that if a DT of a current Q/A pair has a large common sub-tree with a Q/A pair from the positive training set and a significantly smaller one of the negative training set, then this Q/A pair is covered by the *complement* relation (and other way around).

Only RST arcs of the same type of relation (*presentation* relation, such as *Antithesis, subject matter* relation, such as *Condition, and multinuclear* relation, such as *List*) can be matched when computing common sub-trees. We use *N* for a nucleus or situations presented by this nucleus, and *S* for satellite or situations presented by this satellite. *Situations* are propositions, completed actions or actions in progress, and communicative actions and states (including *beliefs, desires, approve, explain, reconcile* and others). Hence we have the following expression for RST-based generalization '^' for two texts $text_1$ and $text_2$:

$text_1 \wedge text_2 = \cup_{i,j} (rstRelation_{1i}, (\ldots,\ldots) \wedge rstRelation_{2j} (\ldots,\ldots))$, where $i \in$ (RST relations in $text_1$), $j \in$ (RST relations in $text_2$). Further, for a pair of RST relations their generalization looks as follows: $rstRelation_1(N_1, S_1) \wedge rstRelation_2 (N_2, S_2) = (rstRelation_1 \wedge rstRelation_2)(N_1 \wedge N_2, S_1 \wedge S_2)$.

The texts in $N_1$, $S_1$ are subject to generalization as phrases. The rules for $rst_1 \wedge rst_2$ are as follows. If $relation\_type(rst_1) != relation\_type(rst_2)$ then similarity is empty. Otherwise, we generalize the signatures of rhetoric relations as sentences: $sentence(N_1, S_1) \wedge sentence(N_2, S_2)$.

## Evaluation

We formed a dataset of Q/A pairs related to car repair recommendations. These pairs were extracted from dialogues as first and second utterance, so that the question is seven – fifteen keywords and answer is three to six sentences. This resource was obtained to train a dialog support system but it also proved to be useful to evaluate search.

This dataset is scraped from (CarPros 2017) and is available at (Github Car Repair Dataset 2017).

Although a number of dialogue-related datasets is available, for the purpose of customer support – specific dialogue we had to collect our own. Our focus was to collect a dataset when a user interacts with an expert solving certain car-related problems.

We provide an example of a sample dialogue:

*$Q_1$: I have checked the spark plugs, replaced the battery, also replaced ignition coils and the solenoid, but I still could not start my car.*
*$A_1$: Is the engine getting spark and fuel? Will it start for a couple seconds using starting fluid?*
*$Q_2$: It did at first but not anymore. What should I do?*
*$A_2$ : Check fuel pressure. But If no pressure check the fuel pump for possible fault. If you have fuel pressure, then using a noid light make sure you have injector pulse. The light should blink when cranking.*
*If light is not blinking does the check engine light come on with key on engine off? If no light then look for a bad main relay or no power to the engine control module.*

In this example, since $Q_1$ includes rhetoric relation of *Contrast* (something is done correctly but still not working), detailed should include *Condition*: it is either done correctly or incorrectly, so both cases need to be addressed (Figures 6 and 7). Correlation between $DT\text{-}Q_1$ and $DT\text{-}A_2$ would be trivial if $Q_1$ does not include a contradiction but instead just formulate a question of the form '*X is not working – how to fix it*?'

To automate the relevance assessment, we considered the dialogue built *correctly* if an *actual* dialogue from the dataset is formed, given the first Q as a seed. Otherwise, if the sequence of utterances does not occur in the dataset, we consider it to be *incorrect*. There are some deficiencies of this approach since some actual dialogs are illogical and some synthetic dialogues built from distinct ones can be plausible, but it allows avoiding a manual assessment. The number of formed answers is limit to three: once initial $Q$ is given, the system forms $A_1$, a set of $A_{2i}$ and $A_{3j}$. $A_1$ is followed by the actual $C_1$ from the dialogue $Q$, so the proper $A_2$ needs to be selected. Analogously, once actual $C_2$ (if applicable) is provided, proper $A_3$ needs to be selected.

As a first baseline approach, we select dialogue construction based on keyword similarity only, without taking into account a dialogue flow by considering a DT-Q. As a second baseline approach, we augment keyword similarity with linguistic relevance by computing maximal common sub- parse trees between the Q and $A_i$.

*Table 1: Correctness of dialogue formation*

| Dialogue type | Q-A | Q-A1-C | Q-A1-C-A2 | Q-A1-C1-A2-C2-A3 |
|---|---|---|---|---|
| Baseline 1 | 62.3±4.5 | 60.2±5.6 | 58.2±5.0 | 52.5±5.7 |
| Baseline 2 | 67.0±4.8 | 63.8±4.8 | 57.3±5.3 | 55.6±5.9 |
| DT-Q dialogue formation | 72.3±5.6 | 70.3±4.9 | 65.1±5.5 | 65.9±5.7 |

For the selected dataset, baseline approach is capable of building correct scenarios in the cases when similar keywords or similar linguistic phrases deliver the only dialogue scenario that is correct. On the contrary, *DT-Q* dialogue formation does not always succeed because some scenarios deviate from actual ones in the training set, although are still plausible. Hence we see 10 and 5% improvement over the first and second baselines respectively for a basic, single-step scenario (Table 1).

As scenario becomes more complex, the chance that the proper scenario is selected by topic relevance decreases. At the same time, overall scenario formation complexity increases, and therefore an error rate for *DT-Q* approach increases as well. For the most complex, 3-step dialogue scenarios, *DT-Q* approach exceeds the baselines by 13 and 10% respectively.

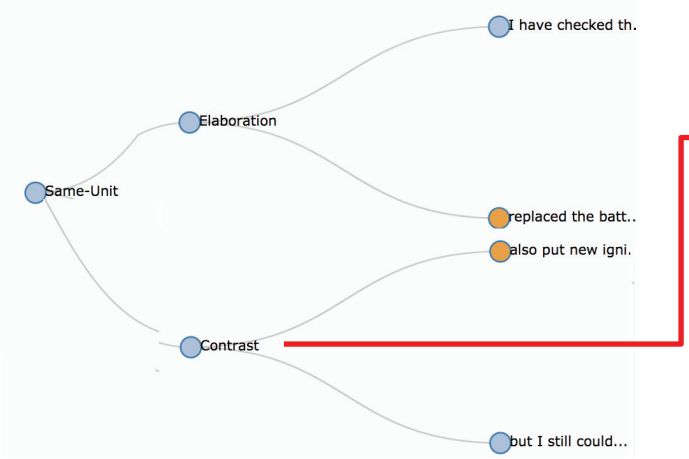


*Fig. 6 Discourse tree for question $Q_1$ in Car Repair domain.*

## Implementation

When a training dataset is indexed, we add a special field to the index with chains of RST relations with selected phrases. Hence candidate answers are parsed and their DTs are built offline. At a conversation time, given a *Q*, the chat bot builds *DT-Q* and forms a query as a chain of RST relations with selected phrases. This query is then run against the special field above as an span-OR query with retained order of RST terms under default TF*IDF relevance. The purpose of such search is to obtain sequences of $Q_i$ candidates.

The component, which enforces complement relation, combines Stanford NLP parsing, coreferences, entity extraction, DT construction (discourse parser, Surdeanu et al 2013 and Joty et al 2016), VerbNet and Tree Kernel builder into one system. The code for this algorithm is available at (Github 2015). Installation instructions, implementation details and integration notes are available at this GitHub page.

The chat bot is optimized to minimize the amount of text the user needs to read in the course of getting to the answer. Instead of reading search results snippets and selecting the ones believed to be relevant, the system suggests the topics related to user question and options to drill into them or chose another one. A chat bot user is expected to read less and interact & clarify more to have a more efficient information access, which is also appropriate for mobile devices where text tends to be shorter and user interaction faster (Fig. 5).

A sample chat bot session is available at (Github 2016), including how topics for clarification are formed and answers are filtered by relevance. A web version of the chat bot is available at (Amazon 2017). A video of a session with a chat bot in financial domain is available at (Oracle 2017).


*Fig. 7 Discourse tree for the detailed answer $A_2$ for $Q_1$*

## Related Systems and Conclusions

Recently, rhetoric parsing became more reliable and efficient (Joty et al 2013, Feng and Hirst 2014); however, the number of applications for resultant discourse trees (DTs) is limited to content generation and summarization. Discourse features are valuable for passage re-ranking (Jensen et al 2014). DTs have been found to assist in answer indexing to make search more relevant: query keyword should occur in nucleus rather than a satellite of a rhetoric relation (Galitsky et al 2015).

The most popular approach in the last few years is to learn topical relevance and dialogue management together, using deep learning. This family of approaches also fall answer category of data-driven (Serban et al 2017); they require huge dialogue datasets.

The problem of reducing the space of possible utterances under dialogue construction has been addressed in the extensive body of research. This reduction is based on syntactic and possibly semantic features, but not discourse ones. A dialogue management system can narrow the number of plausible answer utterances to a small list, and an ML model would select the most appropriate responses. from this list (Lowe et al., 2016). This next utterance classification task is

derived from the IR-based metrics for information-retrieval-based approaches, which is easy to interpret, and tune the complexity by changing the number of false responses.

Modern search engines and chat bots, from vertical to horizontal, do not implement reasoning via discourse - level analysis, to the best of our knowledge. This is due to its computational load and hard compatibility with big data technologies. Most search engineers consider discourse analysis too abstract and too distant from applications.

Since rhetoric parsers for English has become more available and accurate, their application in search engine indexing is becoming more feasible. As precision and recall of search systems ignoring discourse level information deteriorates, users do not find products, services and information they need, leveraging of linguistic technologies including discourse become realistic for industrial systems.

Most chat bot vendors these days such as botframework.com and api.ai provide an NLP platform so that the content providers feed them with Q/A pairs and expect satisfactory performance. It is hard to formally evaluate these systems, but anecdotal evidence is that their performance is rather limited. Another family of chat bots is focused on simulation of intelligent activity of humans instead of providing an efficient content to information. This family is also frequently based on deep learning of a huge set of conversations. Being capable of supporting a conversation on an arbitrary topic, building plausible phrases, these systems are nevertheless hardly applicable for industrial applications such as customer support.

In this study we discovered that a dialogue structure could be built from the discourse tree of an initial question. This structure is built on top of the default conversational structure implementing such features as clarification, personalization or recommendation. If clarification scenario type is chosen, topics are automatically formed by the chat bot and are presented for a user to choose. For personalization, for a user query, the customer support chat bot system reduces the list of resolution scenarios based on what information is available for the given user. Chat bot recommendation scenario proposes a solution to a problem by finding the one accepted by users similar to the current one. Clarification, personalization and recommendation scenario covers only a small portion of plausible customer support scenarios. Discourse analysis of dialogues support dialogue scenario management in a universal way, for a broad range of available text fragments and previously accumulated responses.

## References


Amazon 2017. Financial Chat Bot Demo http://ec2-35-160-183-122.us-west-2.compute.amazonaws.com:8080/ChatBot/indexnew.html

CarPros 2017 http://www.2carpros.com

Feng, WV and Hirst, G. 2014. A linear- time bottom-up discourse parser with constraints and post-editing. In *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, USA, June*.

Joty, SR, Carenini G, Ng RT, and Mehdad Y. 2013. Combining intra-and multi- sentential rhetorical parsing for document-level dis- course analysis. In *ACL (1)*, pages 486–496.

Jansen, P., M. Surdeanu, and Clark P. 2014. Discourse Complements Lexical Semantics for Nonfactoid Answer Reranking. ACL.

Github. 2015. https://github.com/bgalitsky/relevance-based-on-parse-trees.

Github. 2017. Car Repair Dialogue Dataset https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/CarRepairData_AnswerAnatomyDataset2.csv.zip.

Github. 2016. Sample chatbot dialogue. https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/botSessionExample.txt

Chali, Y. Shafiq R. Joty, and Sadid A. Hasan. 2009. Complex question answering: unsupervised learning approaches and experiments. J. Artif. Int. Res. 35, 1 (May 2009), 1-47.

Galitsky, B. 2013. Machine learning of syntactic parse trees for search and classification of text. *Engineering Application of AI* , 26(3) 1072-91.

Galitsky, B, Ilvovsky, D. and Kuznetsov SO. 2015. Rhetoric Map of an Answer to Compound Queries. ACL-2, 681–686.

Galitsky, B., Matching parse thickets for open domain question answering, Data & Knowledge Engineering, Volume 107, January 2017, Pages 24-50

Kipper, K. Korhonen, A., Ryant, N. and Palmer, M. 2008. A large-scale classification of English verbs. Language Resources and Evaluation Journal, 42, pp. 21-40.

Oracle. 2017. Financial Chat Bot Demo. https://drive.google.com/open?id=0B-TymkYCBPsfV3JQSGU3TE9mRVk.

Lowe, R.I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. In Special Interest Group on Discourse and Dialogue.

Mann, W. and Thompson. S. 1988. Rhetorical structure theory: Towards a functional theory of text organization. Text-Interdisciplinary Journal for the Study of Discourse, 8(3):243–281.

Serban, IV, Lowe, R., Henderson, P., Charlin, L., and Pineau J. 2017. A Survey of Available Corpora for Building Data-Driven Dialogue Systems. https://arxiv.org/abs/1512.05742.

Surdeanu, M.,Hicks, T. and Valenzuela-Escarcega MA 2015. Two Practical Rhetorical Structure Theory Parsers. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies: Software Demonstrations.

Wang, W., Su, J., Tan, C.L. 2010. Kernel Based Discourse Relation Recognition with Temporal Ordering Information. *ACL*.

Webscope. 2017. Yahoo! Answers Dataset.

https://webscope.sandbox.yahoo.com/catalog.php?datatype=l.

Wilks, Y. A. (Ed.) 1999. Machine conversations. Kluwer.