A Strategic Reasoning Model for Generating Alternative Answers

Jon Scott Stevens
Center for General Linguistics, Berlin
stevens@zas.gwz-berlin.de

Sebastian Reuße
Ruhr University Bochum
sebastian.reusse@rub.de

Anton Benz
Center for General Linguistics, Berlin

Ralf Klabunde
Ruhr University Bochum

Abstract

We characterize a class of indirect answers to yes/no questions, *alternative answers*, where information is given that is not directly asked about, but which might nonetheless address the underlying motivation for the question. We develop a model rooted in game theory that generates these answers via *strategic reasoning* about possible unobserved domain-level user requirements. We implement the model within an interactive question answering system simulating real estate dialogue. The system learns a prior probability distribution over possible user requirements by analyzing training dialogues, which it uses to make strategic decisions about answer selection. The system generates pragmatically natural and interpretable answers which make for more efficient interactions compared to a baseline.

1 Introduction

In natural language dialogue, questions are often answered indirectly. This is particularly apparent for yes/no questions, where a wide range of responses beyond literal "yes" and "no" answers is available. Sometimes indirect answers serve to anticipate the next step of the hearer's *plan*, as in (1) (Allen and Perrault, 1980), where the literal answer is entailed by the supplied answer, and sometimes indirect answers leave it to the hearer to infer the literal answer from common contextual assumptions, as in (2) (de Marneffe et al., 2009).

(1) Q: Has the train to Windsor left yet?
A: It's leaving soon from gate 7.

(2) Q: Is Sue at work?
A: She's sick with the flu.

But other times there is no semantic link between the question and the supplied answer. Rather, the answer must be interpreted in light of the task-specific goals of the interlocutors. Consider (3) in a context where a customer is posing questions to a real estate agent with the aim of renting an apartment.

(3) Q: Does the apartment have a garden?
A: Well, it has a large balcony.

Whether there is a balcony has no logical bearing on whether there is a garden. Intuitively, the realtor is inferring that the customer's question might have been motivated by a more general requirement (perhaps the customer wants a place to grow flowers) and supplying an alternative attribute to satisfy that requirement. In this case the answerer must reason about which attributes of an apartment might satisfy a customer who would ask about a garden. Note that multiple motivating requirements are possible (perhaps the customer just wants to relax outside), such that the answerer might just as easily have said, "It has a large balcony, and there is a park close by." In either case, the hearer can infer from the lack of a direct answer that the apartment must not have a garden, because if it did, to say so would have been more obviously helpful.

This paper focuses on this class of answers, which we call *alternative answers*. We characterize these as indirect answers to yes/no questions that offer attributes of an object under discussion which might satisfy an unobserved domain-level *requirement* of the questioner. We conceive of a requirement as a set of *satisfying conditions*, such that a particular domain-related need would be met by any one member of the set. For example, in the context of (3) we can encode a possible customer requirement of a place to grow flowers in an apartment, FLOWERS $= \{\text{GARDEN}, \text{BALCONY}\}$, such that *either* GARDEN *or* BALCONY would suffice to satisfy the requirement.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 534–542, Beijing, China, July 26-31, 2015.

In order to generate alternative answers automatically, we must first solve two problems: (i) how does one learn and represent a space of likely user requirements?, and (ii) how does one use such a space to select indirect answers? To do this in a natural, pragmatically interpretable way, we must not only derive answers like in (3), but crucially, also rule out infelicitous responses like the following, where a logically possible alternative leads to incoherence due to the low probability of an appropriate requirement like {GARDEN, BASEMENT}. (In other words, wanting a garden has little effect on the probability of wanting a basement.)

(4) Q: Does the apartment have a garden?
A: #Well, it has a large basement.

To solve these problems, we propose an approach rooted in decision-theoretic and game-theoretic analyses of indirectness in natural language (van Rooij, 2003; Benz and van Rooij, 2007; Benz et al., 2011; Stevens et al., 2014) whereby a system uses *strategic reasoning* to derive an optimal response to a yes/no question given certain domain assumptions. The model operates by assuming that both the questioner and the answerer are *rational*, i.e. that both participants want to further their own goals, and will behave so as to maximize the probability of success at doing so.

One appeal of the strategic approach is its relative simplicity: the model utilizes a learned probability distribution over possible domain-level requirements of the questioner and applies simple probabilistic reasoning to feed content selection during online answer generation. Unlike plan inference approaches, we do not need to represent any complex taxonomies of stimulus conditions (Green and Carberry, 1994) or coherence relations (Green and Carberry, 1999; Asher and Lascarides, 2003).

By implementing the strategic reasoning model within a simple interactive question answering system (Konstantinova and Orasan, 2012), simulating real estate dialogues with exchanges like in (3), we are able to evaluate the current approach quantitatively in terms of dialogue efficiency, perceived coherence of the supplied answers, and ability of users to draw natural pragmatic inferences. We conclude that strategic reasoning provides a promising framework for developing answer generation methods by starting with principled theoretical analyses of human dialogue.

The following section presents the model, including a concrete content selection algorithm used for producing answers to questions, and then walks through a simple illustrative example. Section 3 describes our implementation, addresses the problem of learning requirement probabilities, and presents the results of our evaluation, providing quantitative support for our approach. Section 4 concludes with a general summary.

2 Model

2.1 Overview

We derive our model beginning with a simple description of the discourse situation. In our case, this is an exchange of questions and answers where a *user* poses questions to be answered by an *expert* who has access to a *database* of information that the user wants. The expert has no advance knowledge of the database, and thus must *look up* information as needed. Each user question is motivated by a *requirement*, conceived of as a (possibly singleton) set of database attributes (restricted for current purposes to boolean attributes), any one of which satisfies a user need (e.g. {GARDEN, BALCONY} in the previous section). Only the user has direct access to her own requirements, and only the expert can query the database to inform the user whether her requirements can be satisfied. For current purposes we assume that each question and answer in the dialogue pertains to a specific object o from the database which is designated as the *object under discussion*. This way we can represent answers and question denotations with attributes, like GARDEN, where the queried/supplied attribute is assumed to predicate over o. In these terms, the expert can either ASSERT an attribute (if it holds of o) or DENY an attribute (if it does not hold of o) in response to a user query.

Now we describe the goals of the interlocutors. The user wants her requirements to be satisfied, and will not accept an object until she is sure this is the case. If it is clear that an object cannot satisfy one or more requirements, the user will ask to discuss a different object from the database. We can thus characterize the set of possible user responses as follows: the user may ACCEPT the object as one that meets all requirements, the user may REJECT the object and ask to see something else, or the user may FOLLOW UP, continuing to pose questions about the current object. The user's

goal, then, is ultimately to accept an object that in fact satisfies her requirements, and to reject any object that does not.

The expert's goal is to help the user find an optimal object as efficiently as possible. Given this goal, the expert does better to provide alternative attributes (like BALCONY for GARDEN in (3)) in place of simple "no" answers only when those attributes are relevant to the user's underlying requirements. To use some economic terminology, we can define the *benefit* (B) of looking up a potential alternative attribute a in the database as a binary function indicating whether a is relevant to (i.e. a member of) the user requirement ρ which motivated the user's question. For example, in (3), if the user's question is motivated by requirement {GARDEN, BALCONY}, then the benefit of looking up whether there is a balcony is 1, because if that attribute turns out to hold of o, then the customer's requirement is satisfied. If, on the other hand, the questioner has requirement {GARDEN}, then the benefit of looking up BALCONY is 0, because this attribute cannot satisfy this requirement.

$$B(a|\rho) = 1 \text{ if } a \in \rho \text{ and } 0 \text{ otherwise} \quad (1)$$

Regardless of benefit, the expert incurs a *cost* by looking up information. To fully specify what cost means in this context, first assume a small, fixed effort cost associated with looking up an attribute. Further assume a larger cost incurred when the user has to ask a follow-up question to find out whether a requirement is satisfied. What really matters are not the raw cost amounts, which may be very small, but rather the *relative cost* of looking up an attribute compared to that of receiving a follow-up. We can represent the ratio of look-up cost to follow-up cost as a constant κ, which encodes the reluctance of the expert to look up new information. Intuitively, if κ is close to 1 (i.e. if follow-ups are not much more costly than simple look-ups), the expert should give mostly literal answers, and if κ is close to 0, (i.e. if relative follow-up cost is very high), the expert should look up all potentially beneficial attributes. With this, let the *utility* (U) of looking up a be the benefit of looking up a minus the relative cost.

$$U(a|\rho) = B(a|\rho) - \kappa \quad (2)$$

The expert is utility-maximizing under game-theoretic assumptions, and (assuming a baseline utility of zero for doing nothing) should aim to look up attributes for which U is positive, i.e. for which benefit outweighs cost. But the expert has a problem: ρ, on which U depends, is known only to the user. Therefore, the best the expert can do is to reason probabilistically, based on the user's question, to maximize *expected utility*, or the weighted average of $U(a|\rho)$ for all possible values of ρ. The expected utility of looking up an attribute a can be written as the *expected benefit* of a—the weighted average of $B(a|\rho)$ for all ρ—minus the relative cost. Let REQS be the set of all possible user requirements and let q be the user's question.

$$EU(a|q, \text{REQS}) = EB(a|q, \text{REQS}) - \kappa \quad (3)$$

$$EB(a|q, \text{REQS}) = \sum_{\rho \in \text{REQS}} P(\rho|q) \times B(a|\rho) \quad (4)$$

The probability of a user requirement $P(\rho|q)$ is calculated via Bayes' rule, assuming that users will choose their questions randomly from the set of questions whose denotations are in their requirement set. This yields the following.

$$P(\rho|q) = \frac{P(q|\rho) \times P(\rho)}{\sum_{\rho' \in \text{REQS}} P(q|\rho') \times P(\rho')} \quad (5)$$

$$P(q|\rho) = \frac{1}{|\rho|} \text{ if } [\![q]\!] \in \rho \text{ and } 0 \text{ otherwise} \quad (6)$$

The prior probability of a user requirement, $P(\rho)$, is given as input to the model. We will see in the next section that it is possible to learn a prior probability distribution from training dialogues.

We have now fully characterized the expected benefit (EB) of looking up an attribute in the database. As per Eq.3, the expert should only bother looking up an attribute if EB is greater than the relative cost κ, since that is when EU is positive. The final step is to give the expert a sensible way to iteratively look up attributes to potentially produce multiple alternatives. To this end, we first point out that if an alternative has been found which satisfies a certain requirement, then it no longer adds any benefit to consider that requirement when selecting further alternatives. For example, in the context of example (3), when the realtor queries the database to find the apartment has a balcony, she no longer needs to consider the probability of a requirement {BALCONY, GARDEN} when considering additional attributes, since that is already satisfied. Given this consideration, the order in which database attributes are

looked up can make a difference to the outcome. So, we need a consistent and principled criterion for determining the order in which to look up attributes. The most efficient method is to start with the attribute with the highest possible EB value and then iteratively move down to the next best attribute until EB is less than or equal to cost.

Note that the attribute that was asked about will always have an EB value of 1. Consider again the QA exchange in (3). Recall that the expert assumes that the user's query is relevant to an underlying requirement ρ. This means that ρ must contain the attribute GARDEN. Therefore, by definition, supplying GARDEN will always yield positive benefit. We can use this fact to explain how alternative answers are interpreted by the user. The user knows that the most beneficial attribute to look up (in terms of EB) is the one asked about. If that attribute is not included in the answer, the user is safe to assume that it does not hold of the object under discussion. By reasoning about the expert's reasoning, the user can derive the implicature that the literal answer to her question is "no". In fact, this is what licenses the expert to leave the negation of the garden attribute out of the answer: the expert knows that the user knows that the expert would have included it if it were true. This type of "I know that you know" reasoning is characteristic of game-theoretic analysis.[1]

2.2 Algorithm and example

Our algorithm for generating alternative answers (Algorithm 1), which simulates strategic reasoning by the expert in our dialogue situation, is couched in a simple information state update (ISU) framework (Larsson and Traum, 2000; Traum and Larsson, 2003), whereby the answerer keeps track of the current object under discussion (o) as well as a history of attributes looked up for o (HIST_o). The output of the algorithm takes the form of a dialogue move, either an assertion (or set of assertions) or denial that an attribute holds of o. These dialogue moves can then be translated into natural language with simple sentence templates. The answerer uses HIST_o to make sure redundant alternatives aren't given across QA exchanges. If all possible answers are redundant, the answerer falls back on a direct yes/no response.

[1] It can be shown that the answer selection algorithm presented in this section, combined with a simple user interpretation model, constitutes a *perfect Bayesian equilibrium* (Harsanyi, 1968; Fudenberg and Tirole, 1991) in a signaling game (Lewis, 1969) with private hearer types which formally describes this kind of dialogue.

Requirement set	$P(\rho)$	$P(q\|\rho)$	$P(\rho\|q)$
ρ_G ={GARDEN}	0.5	1	0.67
ρ_F ={GARDEN, BALCONY}	0.25	0.5	0.17
ρ_P ={GARDEN, PARK}	0.2	0.5	0.13
ρ_S ={GARDEN, BASEMENT}	0.05	0.5	0.03

Table 1: A toy example of a customer requirement space with probabilities for q = 'Does the apartment have a garden?'

To illustrate how the algorithm works, consider a simple toy example. Table 1 gives a hypothetical space of possible requirements along with a distribution of priors, likelihoods and Bayesian posteriors. We imagine that a customer might want a garden (ρ_G), or more generally a place to grow flowers (ρ_F), a place for their child to play outside (ρ_P), or, in rare cases, either a garden or a basement to use as storage space (ρ_S). The rather odd nature of ρ_S is reflected in its low prior. Consider a variant of (3) where HIST_o is empty, and where DB_o contains BALCONY, PARK and BASEMENT.

(5) Q: Does the apartment have a garden?
A: It has a balcony, and there is a park very close by.

To start, let REQS contain the requirements in Table 1, and let $\kappa = 0.1$. The algorithm derives the answer as follows. First, the algorithm looks up whether GARDEN holds of o. It does not hold, so GARDEN is not added to the answer; it is only added to the history of looked up attributes.

$a = \text{GARDEN};\ EB(\text{GARDEN}) = 1;$
$\text{HIST}_o = \{\text{GARDEN}\}$

Then, the system finds the next best attribute, BALCONY, which does hold of o, appends it to the answer as well as the history, and removes the relevant requirement from consideration.

$a = \text{BALCONY};\ EB(\text{BALCONY}) = 0.17;$
$\text{HIST}_o = \{\text{GARDEN, BALCONY}\};$
$\text{ANSWER} = \{\text{BALCONY}\};$
$\text{REQS} = \{\rho_G, \rho_P, \rho_S\}$

The attribute PARK is similarly added.

$a = \text{PARK};\ EB(\text{PARK}) = 0.13;$
$\text{HIST}_o = \{\text{GARDEN, BALCONY, PARK}\};$
$\text{ANSWER} = \{\text{BALCONY, PARK}\};$
$\text{REQS} = \{\rho_G, \rho_S\}$

The attribute BASEMENT is next in line. However, its EB value is below the threshold of 0.1 due

Algorithm 1 An algorithm for generating alternative answers

Input: A set of attributes Φ, an object under discussion o, a database DB_o of attributes which hold of o, a history HIST_o of attributes that have been looked up in the database, a set of possible user requirements REQS, a prior probability distribution over REQS, a user-supplied question q with denotation $[\![q]\!]$ and a relative cost threshold $\kappa \in (0, 1)$

Initialize: $\text{ANSWER} = \{\}$; $\text{LOOKUP} = \text{TRUE}$

while LOOKUP **do**
  $\Phi' = (\Phi \setminus \text{HIST}_o) \cup \{[\![q]\!]\}$ ▷ Only consider alternatives once per object per dialogue.
  $a = \arg\max_{\phi \in \Phi'} \text{EB}(\phi | q, \text{REQS})$ ▷ Find the best candidate answer.
  if $\text{EB}(a|q, \text{REQS}) > \kappa$ **then** ▷ Check whether expected benefit outweighs cost.
    $\text{HIST}_o = \text{HIST}_o \cup \{a\}$ ▷ Log which attribute has been looked up.
    if $a \in \text{DB}_o$ **then**
      $\text{ANSWER} = \text{ANSWER} \cup \{a\}$ ▷ Add to answer if attribute holds.
      $\text{REQS} = \text{REQS} \setminus \{\rho \in \text{REQS} \mid \rho \cap \text{ANSWER} \neq \emptyset\}$
      ▷ Don't consider requirements that are already satisfied.
    end if
  else
    $\text{LOOKUP} = \text{FALSE}$ ▷ Stop querying the database when there are no promising candidates left.
  end if
end while
if $\text{ANSWER} \neq \emptyset$ **then** $\text{ASSERT}(\text{ANSWER})$,
else $\text{DENY}([\![q]\!])$
end if

to its low prior probability, and thus the iteration stops there, and BASEMENT is never looked up.

$a = \text{BASEMENT}$; $EB(\text{BASEMENT}) = 0.03$;
$EB < \kappa$; exit loop

3 Implementation and evaluation

3.1 Setup

A simple interactive question answering system was built using a modified version of the PyTrindiKit toolkit[2] with a database back end implemented using an adapted version of PyKE, a Horn logic theorem prover.[3] The system was set up to emulate the behavior of a real estate agent answering customers' yes/no questions about a range of attributes pertaining to individual apartments. A set of 12 attributes was chosen for the current evaluation experiment. The system generates answers by first selecting a discourse move (i.e. assertion or denial of an attribute) and then translating the move into natural language with simple sentence templates like, "It has a(n) X" or "There is a(n) X nearby". When answers are indirect (i.e. not asserting or denying the attribute asked about), the system begins its reply with the discourse connective "well" as in example (3).[4]

Subjects interacted with our system by means of an online text-based interface accessible remotely through a web browser. At the outset of the experiment, subjects were told to behave as if they were finding an apartment for a hypothetical friend, and given a list of requirements for that friend. The task required them to identify which from among a sequence of presented apartments would satisfy the given set of requirements. One out of four lists, each containing three requirements (one of which was a singleton), was assigned to subjects at random. The requirements were constructed by the researchers to be plausible desiderata for users looking for a place to rent or buy (e.g. connection to public transit, which could be satisfied either by a nearby bus stop, or by a nearby train station).

The apartments presented by the system were individually generated for each experiment such that there was an apartment satisfying one attribute for each possible combination of the three requirements issued to subjects, plus two additional apartments that each satisfied two of the conditions ($2^3 + 2 = 10$ apartments overall). Attributes outside a subject's requirement sets were added at random to assess the effect of "unhelpful" alternative answers.

Subject interacted with one of two answer generation models: a *literal* model, which only produced direct yes/no answers, and the *strategic*

[2] `https://code.google.com/p/py-trindikit`

[3] `http://pyke.sourceforge.net/`

[4] Early feedback indicated that alternative answers were more natural when preceded by such a discourse connective. To assess this effect, we ran a separate evaluation experiment with an earlier version of the system that produced alternative answers without "well". Dialogue lengths and coherence scores were not very different from what is reported in this section; however, in contrast with the current evaluation, we found a large effect of model type (a 69% decrease for strategic vs. literal) on whether the subjects successfully completed the task (z=-2.19, p=0.03). This is consistent with the early feedback.

model as outlined above. Crucially, in both conditions, the sequence in which objects were presented was fixed so that the last apartment offered would be the sole object satisfying all of the desired criteria. Also, we set the strategic model's κ parameter high enough ($1/7$) that only single-attribute answers were ever given. These two properties of the task, taken together, allow us to obtain an apples-to-apples comparison of the models with respect to average dialogue length. If subjects failed to accept the optimal solution, the interaction was terminated. After completing interaction with our system, subjects were asked to complete a short survey designed to get at the perceived coherence of the system's answers. Subjects were asked to rate, on a seven-point Likert scale, the relevance of the system's answers to the questions asked, overall helpfulness, the extent to which questions seemed to be left open, and the extent to which the system seemed evasive.

We predict that the strategic system will improve overall efficiency of dialogue over that of the literal system by (i) offering helpful alternatives to satisfy the customer's needs, and (ii) allowing customers to infer implicit "no" answers from alternative answers, leading to rejections of sub-optimal apartments. If, contrary to our hypothesis, subjects fail to draw inferences/implicatures from alternative answers, then we expect unhelpful alternatives (i.e. alternatives not in the user's requirement set) to prompt repeated questions and/or failures to complete the task.

With respect to the questionnaire items, the literal system is predicted to be judged maximally coherent, since only straightforward yes/no answers are offered. The question is whether the pragmatic system also allows for coherent dialogue. If subjects judge alternative answers to be incoherent, then we expect any difference in average Likert scale ratings between strategic and literal system to reflect the proportion of alternative answers that are given.

3.2 Learning prior probabilities

Before presenting our results, we explain how prior probabilities can be learned within this framework. One of the assumptions of the strategic reasoning model is that users ask questions that are motivated by specific requirements. Moreover, we should assume that users employ a reasonable questioning strategy for finding out whether

S:	An apartment in the north of town might suit you. I have an additional offer for you there.
U:	Does the apartment have a garden?
S:	The apartment does not have a garden.
U:	Does the apartment have a balcony?
S:	The apartment does not have a balcony.
U:	I'd like to see something else

Figure 1: An example of the negation-rejection sequence $\langle$GARDEN, BALCONY$\rangle$

requirements hold, which is tailored to the system they are interacting with. For example, if a user interacts with a system that only produces literal yes/no answers, the user should take all answers at face value, not drawing any pragmatic inferences. In such a scenario, we expect the user's questioning strategy to be roughly as follows: for $a_1, a_2, \cdots, a_n$ in requirement ρ, ask about a_1, then if a_1 is asserted, accept (or move on to the next requirement if there are multiple requirements), and if not, ask about a_2; if a_2 is asserted, accept, and if not, ask about a_3, and so on, until a_n is asked about. If a_n is denied, then reject the object under discussion. If you need a place to grow flowers, ask if there is a balcony or garden, then, if the answer is no, ask about the other attribute. If no "yes" answers are given, reject.

Such a strategy predicts that potential user requirements should be able to be gleaned from dialogues with a literal system by analyzing *negation-rejection sequences* (NRSs). A negation-rejection sequence is a maximal observed sequence of questions which all receive "no" answers, without any intervening "yes" answers or any other intervening dialogue moves, such that at the end of that sequence of questions, the user chooses to reject the current object under discussion. Such a sequence is illustrated in Fig.1. By hypothesis, the NRS $\langle$GARDEN, BALCONY$\rangle$ indicates a possible user requirement $\{$GARDEN, BALCONY$\}$.

By considering NRSs, the system can learn from training data a reasonable prior probability distribution over possible customer requirements. This obviates the need to pre-supply the system with complex world knowledge. If customer requirements can in principle be learned, then the strategic approach could be expanded to dialogue situations where the distribution of user requirements could not sensibly be pre-supplied. While the system in its current form is not guaranteed to scale up in this way, its success here provides us with a promising proof of concept.

Using the dialogues with the literal system as training data, we were able to gather frequencies of observed negation-rejection sequences. By transforming the sequences into unordered sets and then normalizing the frequencies of those sets, we obtained a prior probability distribution over possible customer requirements. In the training dialogues, subjects were given the same lists of requirements as was given for the evaluation of the strategic model. If successful, the system should use the yes/no dialogue data to learn high probabilities for requirements which customers actually had, and low probabilities for any others, allowing us to evaluate the system without giving it any prior clues as to which customer requirements were assigned.

Because we know in advance which requirements the subjects wanted to fulfill, we have a gold standard against which we can compare the question-alternative answer pairs that different variants of the model are able to produce. For example, we know that if a subject asked whether the apartment had a balcony and received an answer about a nearby café, that answer could not have been beneficial, since no one was assigned the requirement $\{\text{CAFÉ}, \text{BALCONY}\}$.

Table 2 compares three variant models: (i) the system we use in our evaluation, which sets prior probabilities proportional to NRS frequency, (ii) a system with flat priors, where probability is zero if NRS frequency is zero, but where all observed NRSs are taken to correspond to equiprobable requirements, and finally (iii) a baseline which does not utilize an EB threshold, but rather simply randomly selects alternatives which were observed at least once in an NRS with the queried attribute. These models are compared by the maximum benefit of their possible outputs using best-case values for κ. We see that there is a good match between the answers given by the strategic model with learned priors and the actual requirements that users were told to fulfill.

Though it remains to be seen whether this would scale up to more complex requirement spaces, this result suggests that NRSs can in fact be indicative of disjunctive requirement sets, and can indeed be useful in learning what possible alternatives might be. For purposes of our evaluation, we will see that the method was successful.

Model	Precision	Recall	F1
Frequency-based	1	0.92	0.96
Flat	0.88	0.92	0.90
Baseline	0.23	1	0.37

Table 2: Comparison of best-case output with respect to potential benefit of alternative answer types to subjects. Precision = hits / hits+misses, and Recall = hits / possible hits. A "hit" is a QA pair which is a possible output of the model, such that A could be a beneficial answer to a customer asking Q, and a "miss" is such a QA pair such that A is irrelevant to Q.

3.3 Evaluation results

We obtained data from a total of 115 subjects via Amazon Mechanical Turk; 65 subjects interacted with the literal comparison model, and 50 subjects interacted with the strategic model. We excluded a total of 13 outliers across both conditions who asked too few or too many questions (1.5 interquartile ranges below the 1st or above the 3rd quartile). These subjects either quit the task early or simply asked all available questions even for apartments that were obviously not a good fit for their requirements. Two subjects were excluded for not filling out the post-experiment questionnaire. This left 100 subjects (59 literal/41 strategic), of which 86 (49/37) successfully completed the task, accepting the object which met all assigned requirements. There was no statistically significant difference between the literal and strategic models with respect to task success.

We first compare the literal and strategic models with regard to dialogue length, looking only at the subjects who successfully completed the task. Due to the highly structured nature of the experiment it was always the case that a successful dialogue consisted of 10 apartment proposals, some number of QA pairs, where each question was given a single answer, 9 rejections and, finally, one acceptance. This allows us to use the number of questions asked as a proxy for dialogue length. Figure 2 shows the comparison. The strategic model yields 27.4 questions on average, more than four fewer than the literal model's 31.6. Standard statistical tests show the effect to be highly significant, with a one-way ANOVA yielding F=16.2, p = 0.0001, and a mixed effects regression model with a random slope for item (the items in this case being the set of requirements assigned to the sub-

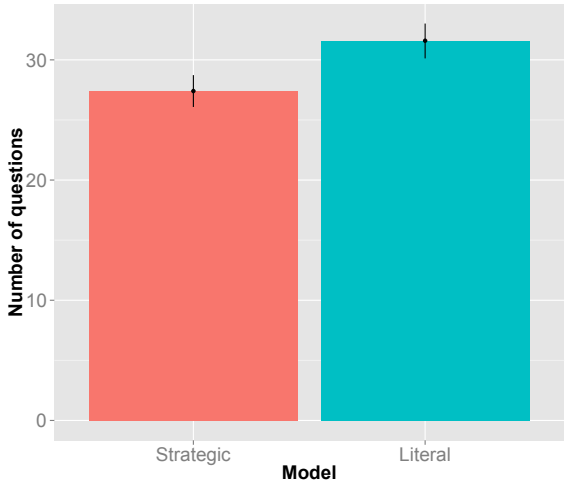

Figure 2: Avg. number of QA pairs by model

S:	How about an apartment in the east of the city? I have an offer for you there.
U:	Does the apartment have a café nearby?
S:	Well, there is a restaurant nearby.
U:	I'd like to see something else

Figure 3: A QA exchange from a dialogue where the user was instructed to find an apartment with a café nearby

ject) yielding t=4, p=0.0001.

We now ask whether the observed effect is due only to the presence of helpful alternatives which preclude the need for follow-up questions, or whether the ability of users to draw pragmatic inferences from *unhelpful* alternatives (i.e. alternatives that don't actually satisfy the user's requirement) also contributes to dialogue efficiency. Figure 3, taken from a real dialogue with our system, illustrates such an inference. The subject specifically wants a café nearby, and infers from the alternative answer that this requirement cannot be satisfied, and therefore rejects. The subject could have asked the question again to get a direct answer, which would have had a negative effect on dialogue efficiency, but this did not happen. We want to know if subjects' aggregate behavior reflects this example.

First, take the null hypothesis to be that subjects do *not* reliably draw such negative implicatures. In that case we would expect a certain proportion of questions to be repeated. Subjects are allowed to ask questions multiple times, and alternatives are never presented twice, such that repeating questions will ultimately lead to a direct yes/no answer. We do see some instances of this behavior in the

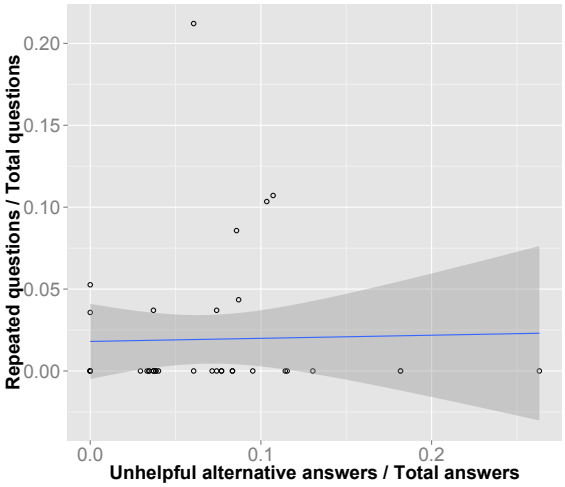

Figure 4: Proportion unhelpful alternatives vs. proportion repeated questions

dialogues. If this is indicative of an overall difficulty in drawing pragmatic inferences from an online dialogue system, then we expect the number of such repetitions to reflect the number of unhelpful alternatives that are offered. Instead, we find that when we plot a linear regression of repeated questions vs. unhelpful alternatives, we get a flat line with no observable correlation (Fig.4). Moreover, we also find no effect of unhelpful alternatives on whether the task was successfully completed. This suggests that the correct inferences are being drawn, as in Fig.3.

We now look at the perceived coherence of the dialogues as assessed by our post-experiment questionnaire. We obtain a composite *coherence score* from all coherence-related items on the seven point Likert scale by summing all per-item scores for each subject and normalizing them to a unit interval, where 1 signifies the upper bound of perceived coherence. Although there is a difference in mean coherence score between the strategic and literal models, with the strategic model exhibiting 88% perceived coherence and the literal model 93%, the difference is not statistically significant. Moreover, we can rule out the possibility that the strategic model is judged to be coherent only when the number of alternative answers is low. To rule this out, we calculate the expected coherence score under the null hypothesis that coherence is directly proportional to the proportion of literal answers. Taking the literal model's average score of 0.93 as a ceiling, we multiply this by the proportion of literal answers to obtain a

null hypothesis expected score of about 0.75 for the strategic model. This null hypothesis is disconfirmed ($F=12.5$, $t=30.6$, $p<0.01$). The strategic model is judged, by the criteria assessed by our post-experiment questionnaire, to be pragmatically coherent independently of the rate of indirect answers given.

4 Conclusion

We have characterized the class of alternative answers to yes/no questions and proposed a content selection model for generating these answers in dialogue. The model is based on strategic reasoning about unobserved user requirements, and is based on work in game-theoretic pragmatics (Benz and van Rooij, 2007; Stevens et al., 2014). The model was implemented as an answer selection algorithm within an interactive question answering system in a real estate domain. We have presented an evaluation of this system against a baseline which produces only literal answers. The results show that the strategic reasoning approach leads to efficient dialogues, allows pragmatic inferences to be drawn, and does not dramatically reduce the overall perceived coherence or naturalness of the produced answers. Although the strategic model requires a form of world knowledge—knowledge of possible user requirements and their probabilities—we have shown that there is a simple method, the analysis of negation-rejection sequences in yes/no QA exchanges, that can be used to learn this knowledge with positive results. Further research is required to address issues of scalability and generalizability, but the current model represents a promising step in the direction of pragmatically competent dialogue systems with solid basis in formal pragmatic theory.

Acknowledgments

This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) (Grant nrs. BE 4348/3-1 and KL 1109/6-1, project 'Pragmatic Requirements for Answer Generation in a Sales Dialogue'), and by the Bundesministerium für Bildung und Forschung (BMBF) (Grant nr. 01UG0711).

References

James F. Allen and C. Raymond Perrault. 1980. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178.

N. Asher and A. Lascarides. 2003. *Logics of Conversation*. Studies in Natural Language Processing. Cambridge University Press.

Anton Benz and Robert van Rooij. 2007. Optimal assertions, and what they implicate. a uniform game theoretic approach. *Topoi*, 26(1):63–78.

Anton Benz, Nuria Bertomeu, and Alexandra Strekalova. 2011. A decision-theoretic approach to finding optimal responses to over-constrained queries in a conceptual search space. In *Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue*, pages 37–46.

Marie-Catherine de Marneffe, Scott Grimm, and Christopher Potts. 2009. Not a simple yes or no. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 136–143.

Dan Fudenberg and Jean Tirole. 1991. Perfect Bayesian equilibrium and sequential equilibrium. *Journal of Economic Theory*, 53(2):236–260.

Nancy Green and Sandra Carberry. 1994. Generating indirect answers to yes-no questions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 189–198.

Nancy Green and Sandra Carberry. 1999. Interpreting and generating indirect answers. *Computational Linguistics*, 25(3):389–435.

John C. Harsanyi. 1968. Games of incomplete information played by 'Bayesian' players, part II. *Management Science*, 14(5):320–334.

Natalia Konstantinova and Constantin Orasan. 2012. Interactive question answering. *Emerging Applications of Natural Language Processing: Concepts and New Research*, pages 149–169.

Staffan Larsson and David R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3&4):323–340.

David Lewis. 1969. *Convention: A Philosophical Study*. Cambridge University Press, Cambridge.

Jon Scott Stevens, Anton Benz, Sebastian Reuße, Ronja Laarmann-Quante, and Ralf Klabunde. 2014. Indirect answers as potential solutions to decision problems. In *Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue*, pages 145–153.

David R. Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In Jan van Kuppevelt and Ronnie W. Smith, editors, *Current and new directions in discourse and dialogue*, pages 325–353. Springer.

Robert van Rooij. 2003. Questioning to resolve decision problems. *Linguistics and Philosophy*, 26(6):727–763.