Does this answer your Question? Towards Dialogue Management for Restricted Domain Question Answering Systems

Matthias Denecke and Norihito Yasuda
Communication Science Laboratories
NTT Corporation
Seika-Cho, Hikaridai, Kyoto, Japan
{denecke,n-yasuda}@cslab.kecl.ntt.co.jp

Abstract

The main problem when going from task-oriented dialogue systems to interactive restricted-domain question answering systems is that the lack of task structure prohibits making simplifying assumptions as in task-oriented dialogue systems. In order to address this issue, we propose a solution that combines representations based on keywords extracted from the user utterances with machine learning to learn the dialogue management function. More specifically, we propose to use Support Vector Machines to classify the dialogue state containing the extracted keywords in order to determine the next action to be taken by the dialogue manager. Much of the content selection for clarification question usually found in dialogue managers is moved to an instance-based generation component. The proposed method has the advantage that it does not rely on an explicit representation of task structure as is necessary for task oriented dialogue systems.

1 Introduction

Question answering is the task of providing natural language answers to natural language questions using an information retrieval engine. Due to the unrestricted nature of the problem, shallow and statistical methods are paramount. Question answering systems work well in the presence of highly specific keywords in the queries, but deteriorate in the case of vague or ambiguous questions.

Dialogue systems address the problem of accessing information from a structured database (such as time table information) or controlling appliances by voice. Due to the fact that the scope of the application defined by the back-end, the domain of the system is well-defined.[1] Therefore, in the presence of vague, ill-defined or misrecognized input from the user, dialogue management, relying on the domain restrictions as given by the application, can interactively request more information from the user until the users' intent has been determined.

1.1 Problem addressed in this paper

Restricted domain question answering systems can be deployed in interactive problem solving solutions, for example, software trouble shooting. An overview of potential applications is given in (Minock, 2005). In these scenarios, interactivity becomes a necessity. This is because it is highly unlikely that all facts relevant to retrieving the appropriate response are stated in the query. For example, in the software trouble shooting task described in (Kiyota et al., 2002), a frequent system generated information seeking question is for the version of the software. Therefore, there is a need to inquire additional problem relevant information from the user, depending on the interaction history and the problem to be solved.

In this paper, we address the problem to learn the function of dialogue management in restricted domain question answering systems. Learning such a function is interesting in the context of task oriented spoken dialogue managers, because dialogue management in the presence of imprecise information, such confidence scores from recognizers, can be optimized. However, when applied to restricted domain question answering systems, learning dialogue management becomes even more appealing. This is because decisions that are comparatively simple for task-oriented dialogue systems, such as deciding whether to end the dialogue, are difficult for interactive restricted domain question answering systems. The reason for this is that dialogue management decisions need to be based on the context, and the retrieval results from the question answering system *only*. In other

[1] This fact poses problems on its own, such as how to make the user understand the scope of the application.

words, in our situation, we cannot take recourse to a task structure telling us that all slots are filled, and the dialogue can be ended.

1.2 Our Approach

Our central idea to dialogue management for restricted domain question answering systems is to defer much of the work to the natural language generation component. The dialogue manager is only responsible for choosing one among a set of predefined actions. (The equivalent for a task for a task oriented dialogue manager would be just to determine whether to prompt for a new slot filler or to confirm a filled slot, but without deciding which slot to prompt or to confirm.) The concrete realization of the action to be taken is carried out by an instance-based generation component that modifies example sentences from that chosen class to the given context and presents them to the user. The generation component is described in detail in Denecke and Tsukada (2005).

The motivation for our approach is twofold. First, as discussed in more detail below, the back-end of an interactive restricted domain question answering system provides much less structure than that of a task oriented dialogue system. This is due to the absence of database schemata. Therefore, it becomes impossible in our setting to make simplifying assumptions as can be done in the context of task-oriented dialogue systems. Instead, we propose to overcome the lack of structure primarily by using instance-based generation, where task-related information is represented implicitly in the example dialogues based on which generation takes place.

Second, we need to address the problem of data sparseness. It is difficult to obtain sufficient data such that a complex dialogue function can be learned reliably (see also section 3). For this reason, we intend to reduce the complexity of dialogue management, and assign some of the tasks usually performed by a dialogue manager to the generation component. The focus of this paper is how best to learn a classifier that chooses the appropriate action.

Since the action to be taken depends both on the dialogue context and the retrieval results of the question answering system, both need to be taken into account by the learning algorithm. More specifically, we are interested in learning a function

$$a = f(c, l)$$

that, given a dialogue context c and an n-best list of retrieved documents l, decides an appropriate action a for the dialogue manager. In particular, we wish to learn a function that answers the two questions:

1. Does the user need to be prompted for more information?

2. If so, how should the user be prompted?

We view the problem as a multiclass classification problem, that is, we attempt to classify the tuple $\langle c, l \rangle$ into a set of classes each of which corresponds to an action. For classification, we use Support Vector Machines (Vapnik, 1995). We compare the efficacity of several kernel functions. Levin et al (2000) argue that supervised learning is not appropriate for optimizing a dialogue strategy. We point out, however, that our goal is not the optimiztation of a dialogue strategy. Instead, our goal is to determine whether the current dialogue context is lacking information, and if so, how it best can be obtained, given past experience.

To summarize, our approach can be described as follows. After processing the current user utterance, a multi-class classifier assigns one out of a few labels to the current dialogue state. The label constrains the form of action to be generated by the dialogue manager. Subsequently, an instance-based generation component retrieves an utterance from a corpus that is similar to the one to be generated. The retrievd utterance is adapted to the current context by replacing content words and presented to the user.

1.3 Our system

We implemented an interactive restricted domain question answering system that combines features of question answering systems with those of spoken dialogue systems. We integrated the following two features in our system: (1) As in question answering systems, the system draws its knowledge from a database of unstructured text. (2) As in spoken dialogue systems, the system can interactively query for more information in the case of vague or ill-defined user queries.

We address the case of restricted domain systems by which we mean the restriction of the systems' scope to one large domain, such as tourism, finance or the like. Typically, this results in a system that, from the standpoint of vocabulary size, is located somewhere between question answering systems and spoken dialogue systems. It is large enough that domain modeling as done in spoken dialogue systems typically becomes inpractical due to the required manual labor, but not large enough to answer all questions one might think of.

We show an overview of our system in figure 1. In particular, the relationship between the question answering system and the dialogue manager can be seen. The user input is passed on to the question

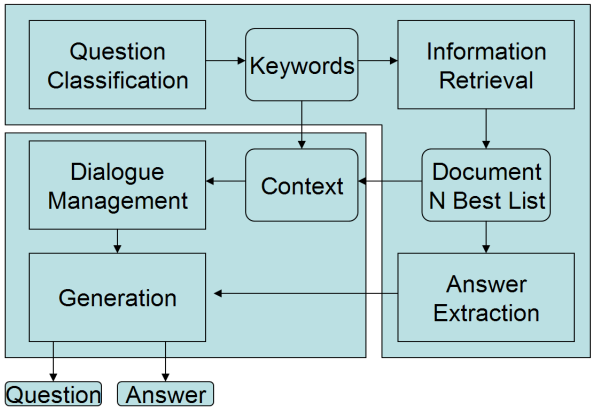

Figure 1: Overview of the system.

answering system which extracts keywords and classifies the question. The determined information is integrated with the context and passed on to the information retrieval engine which returns an n-best list of newspaper articles that best fit the question. Subsequently, the dialogue manager needs to determine whether to present the answer extracted from the highest ranking document to the user or whether to inquire additional information from the user.

2 Background

Task-oriented dialogue systems were among the first commercially available natural language processing systems. The manageability of task-oriented systems stems from the fact that natural language processing, when limited to a specific task, can impose several simplifying assumptions. Thus, the natural language processing aspect of it becomes more manageable.

However, the simplifying assumptions are not limited to the size of grammars or the possibility to use template based natural language generation. Dialogue management proper benefits from the fact that the purpose of the dialogue is the interaction of the user with a back-end system with limited functionality. Interactive question answering systems whose back-end application is a generic information retrieval system do not have the benefit of structure in the way task oriented dialogue systems have.

2.1 Structured and Unstructured Information for Dialogue Processing

In this section, we point out the ways in which the structure of the back-end application has been taken advantage of in dialogue managers. In the following section, we discuss diferent approaches how the first available prototypes of interactive question answering systems try to compensate for the lack of structure in the back-end application.

Given the nature of the application, database schemata of SQL databases are one of the most often used sources of structured information in dialogue managers. A database schema can be seen as a form of type information on the information stored in databases, much the same as XML Schema is a typing scheme for XML documents. For example, Ferrieux and Sadek (1994) describe a way to determine which slots to prompt the user based on the retrieved records from a database and the schema. It is the structure of the database, as represented in the schema, together with the number of potential fillers, that determines what values to prompt. Similarly, Denecke and Waibel (1997) propose a data driven approach to determining which questions to ask based on the structure of an ontology. The difference to the work by Ferrieux and Sadek is that the ontology adds structure to the retrieved record sets so that cascading follow-up questions, leading to the filler of only one slot, can be asked. Also, using subsumption information, potential fillers can be inferred from the ontology in case database access is not possible (for example because the result set is too large). Flycht-Eriksson and Jönsson (2003) propose to use a domain specific ontology to enhance the information contained in free form text. A different problem was tackled by Rudnicky and Wu (1999) who introduced task models to dialogue management. Here, it is the structure of the domain that guides the sort of questions that are asked. A task model is a hierarchical representation of tasks that make up the dialogue. At any point, one task is active. Depending on the interaction with the user, the system moves to another task, a subtask, or repeats the current task. In all three examples, task specific structure helps to answer the two questions outlined in section 1.2.

2.2 Question Answering Systems and Dialogue Systems

One of the earliest systems in which a free text database could be queried in natural language was proposed by Wilensky et al (1984). The queried text consisted of the Unix manual pages, and while the system had limited dialogue capabilities, contextual information could be processed.

More recently, Kiyota et al (2002) proposed an interactive question answering system that helps users troubleshoot problems with computer systems. In case the user does not present all information necessary to determine the correct help text, a dialogue manager detects vague questions and, if necessary, prompts the user for additional information. The prompting is based on so-called *dialogue cards* which can be seen as simplified dialogue scripts. If the ques-

System name	How to decide			Structure
	Whether to ask a question?	What to ask?	How to Generate?	
Dialog Navigator	Match against list of questions	Dialogue cards Information gain (New version)	Dialogue cards	Handcoded
Hitiqa	Conflicts, missing fillers in frames, scoring	Conflicts, missing fillers in frames	Templates?	Extracted WordNet
Spiqa	Scoring	Wh question, specializing ambiguous phrase	Templates	Extracted

Table 1: Features of implemented interactive open domain question answering systems

tion of the user matches one of a list of questions on the dialogue card, predetermined information seeking questions associated with the dialogue card are presented to the user. However, the approach using dialogue cards cannot be easily extended to an open-domain interactive question answering system since the cost of creating the dialogue cards would be prohibitive.

In an updated version of the system, an information gain criterium is proposed to decide which question to ask (Misu and Kawahara, 2005). As in the case of the dialogue cards, a set of pre-prepared questions is provided.

One approach to determine whether to pursue a dialogue is to match the question with the retrieved answer. In case the match is poor, it is assumed that the information provided by the user is not specific enough; therefore the system engages in dialogue. Matching between questions and answers, albeit with a different motivation, is described in Brill et al (2002). In order to avoid question classification, (Brill et al., 2002) proposes to reformulate the query into a declarative sentence and to rank documents retrieved by a search engine based on whether (and how often) a given document contains a sentence with similar structure. The query reformulation takes places *en lieu* of question classification.

Hori et al (2003) propose an interactive voice question answering system in which a list of information seeking questions is hypothesized for each user input. The information seeking questions are generated by using templates in which chunks of the user question are inserted. The template depends on the type of the question as determined by the question answering system. For each hypothesized question, an ambiguity score is calculated. This score depends on the result set and the phrase inserted in the template. If there is a disambiguation question with a score higher than a given threshold, the question is asked; otherwise the answer is generated.

In order to increase the structure of the documents returned from the question answering system, Small et al (2004) propose to extract information using various means from the retrieved documents. The extracted information is then represented in frame-like structures which allows the application of methods known from task-oriented dialogue management.

The work on interactive question answering systems cited above and proposed here is different from the work proposed in DeBoni and Manadhar (2003) in that here, the authors are interested in detecting whether a question from a user is a follow-up question to a previous question or not. This contrasts with our desire to determine the need for system initiated clarification dialogues and the generation thereof. Of course, from an interaction perspective, the differences in approaches result in different degrees of system initiative, but the underlying technologies are also different.

2.3 Support Vector Machines

In recent years, Support Vector Machines have been applied successfully in several pattern recognition applications (Vapnik, 1995). Their popularity is due to the fact that they can be applied in high dimensional feature spaces without actually incurring the high cost of explicitly computing the feature map. Support Vector Machines are instances of supervised learning algorithms. A supervised learning algorithm attempts to learn a decision function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from labeled examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. In the case of Support Vector Machines, we have $\mathcal{Y} = \{-1, +1\}$.

Given a set of labeled training examples, a Support Vector Machine attempts to determine hyperplanes separating the positive from the negative examples such that the margin between the classes is maximized. In order to improve classification performance, the training examples are separated not in the input space but in some high-dimensional feature space. The reason this can be done efficiently is that instead of determining the image $\phi(\mathbf{x}_i)$ of the training examples in the feature space and calculating their inner product $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ there, the distance is calculated implicitly by the *Kernel function* $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

2.3.1 Multi-Class Classification

As described in section 1.2, we would like to classify the dialogue and the user input into one out of n several classes. The classification is used to determine what kind of output to the user is to be generated. Section 3 gives an overview how the number and meaning of the classes is determined.

However, standard Support Vector Machines provide only binary classification. There are several proposals how to do multi-class classification using Support Vector Machines. *One-vs-all* classification is a simple approach in which one binary classifier is trained separately for each class, whereby the members of that class are given as positive examples and members of all other classes are given as negative examples. During classification, each binary classifier output a distance of the input to the closest decision boundary. This distance can be used as an indicator "how sure" the classifier is about the classification. Therefore, the classifier outputting the largest positive number wins, and determines the class of the input.

An alternative is to follow the approach of Weston and Watkins (1999), where a form of voting takes place during classification.

2.3.2 Convolution Kernels

Initially, kernel-based learning algorithms have been applied to various tasks in attribute-value learning in which attributes are represented as components x_i vector $\mathbf{x}$. This approach does not scale well to domains such as natural language processing in which the representation of structure is necessary in order to achieve good classification performance. While *Bag-of-Words* techniques can be employed as an approximation to derive features for classifiers, the loss of structure is not desirable. To address this problem, Haussler (1999) proposed *Convolution Kernels* that are capable of processing structured objects x and y consisting of components $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$. The convolution kernel of x and y is given by the sum of the products of the components' convolution kernels. This approach can be applied to structured obejct of various kinds, and results have been reported for string kernels and tree kernels.

The idea behind Convolution Kernels is that the kernel of two structures is defined as the sum of the kernels of their parts. Formally, let D be a positive integer and $X, X_1, \ldots, X_D$ separable metric spaces. Furthermore, let x and y be two structured objects, and $\mathbf{x} = x_1, \ldots, x_D$ and $\mathbf{y} = y_1, \ldots, y_D$ their parts. The relation $R \subseteq X_1 \times \ldots \times X_D \times X$ holds for $\mathbf{x}$ and x if $\mathbf{x}$ are the parts of x. The inverse R^{-1} maps each structured object onto its parts, i.e. $R^{-1}(x) = \{\mathbf{x} : R(\mathbf{x}, x)\}$. Then the kernel of x and y is given by the following generalized convolution:

$$K(x, y) = \sum_{\mathbf{x} \in R^{-1}(x)} \sum_{\mathbf{y} \in R^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d)$$

Informally, the value of a convolution kernel for two objects X and Y is given by the sum of the kernel value for each of the substructures, i.e. their convolution.

Collins and Duffy (2001) described the application of Convolution Kernels to several natural language processing tasks, focussing on the case where the substructures convoluted by the kernel are trees.

Suzuki et al (2003) proposed *Hierarchical Directed Acyclic Graph* kernels in which the substructures contain nodes which can contain graphs themselves. The hierarchy of graphs allows extended information from multiple components to be represented and used in classification. In addition, nodes may be annotated with attributes, such as part of speech tags, in order to add information. For example, in a Question-Answering system, components such as Named Entity Extraction, Question Classification, Chunking and so on may each add to the graph. Figure 2 shows the graph structure of a sentence after having been preprocessed; this graph structure is the structure that is processed by the HDAG kernel.

The way to arrive at the graph structures (and that we follow) is to subject each sentence to an extensive analysis. Since in Japanese, words are not separated by spaces, the input sentence is broken in words and analyzed morphologically by Chasen (Asahara and Matsumoto, 2000). Based on this analysis, chunking is done using CaboCha (Kudo and Matsumoto, 2002). Each node with which a word stem is associated with semantic information extracted from "Goi-Taikei" (Ikehara et al., 1997) a semantic network similar to the English WordNet.

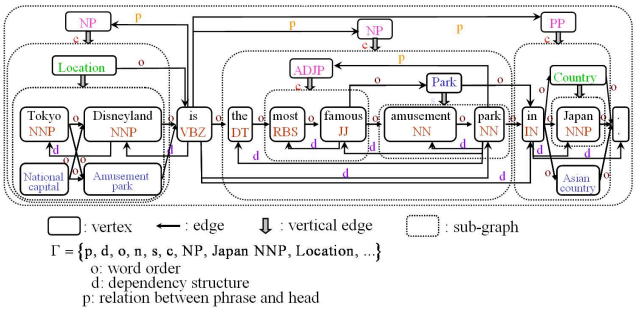

Figure 2: An example.

3 Corpus

We collected a corpus for our instance based generation system as follows. We set up communications

between a wizard and users. The wizard was instructed to "act like the system" we intend to build, that is, she was required to interact with the user either by prompting for more information or give the user the information she thought he wanted.

3.1 Data Collection

The subjects were instructed to ask travel-related questions. The questions the subjects were instructed to ask comprise what kind of accommodation is available, if the destination is known for special food (various areas in Japan are renowned for food that can be had only there), famous sightseeing spots and events, if the destination is important for the history of Japan, and if so, how. The restriction of the kind of questions was motivated by the fact that certain kinds of questions are not well suited for interactive restricted domain question answering systems. In particular, simple factual questions (such as: "How much is a stay in the hotel for three nights?") are better handled by task oriented dialogue systems, a problem we did not want to address in this study. Altogether, 20 users participated in the data collection effort. The participating users were no computer experts, and did not participate in similar resarch experiments before.

3.2 Corpus Properties

Each of the 20 users contributed 8 to 15 dialogues. The length of the dialogues varies between 11 and 84 turns, the median being 34 turns. Altogether, the corpus consists of 201 dialogues. The corpus consists of 6785 turns, 3299 of which are user turns and the remaining 3486 are wizard turns. Due to the strict dialogue regiment prescribed in the onset of the data collection, each dialogue consists either of an equal number of user and wizard turns (in case the user ends the dialogue; 14 cases) or one wizard turn more than user turn in case the wizard ends the dialogue (187 cases). Figure 3 shows the first part of a dialogue from the corpus.

Due to the nature of the Japanese language, phenomena interesting to dialogue processing, such as ellipsis and anaphora, are mostly realized through omission (implicit contextual understanding is often required in Japanese). Due to the strict regimen in which the data was collected, misunderstandings were rare, and corrections did not occur often.

3.3 Corpus Annotation

We recall from section 1.3 that the purpose of the system-generated questions is to integrate additional information from the user in the information retrieval. The corpus contains questions with differing specificity. For generation purposes, it is necessary to control the degree of vagueness. Therefore, we annotate the corpus accordingly. To do that, each dialogue is divided in utterance pairs. Each utterance pair contains a user turn followed by a wizard turn. Each utterance pair is manually classified into one out of 8 classes according to the following schema (*ut* and *wt* refer to *user utterance* and *wizard utterance*, respectively):

```
if (wt is not question)
    if (ut is yes/no question) assign 1
    else if (ut is enumeration question) assign 2
    else if (ut is wh question) assign 3
    else assign 4
else
    if (wt is yes/no question) assign 5
    else if (wt is enumeration question) assign 6
    else if (wt is wh question) assign 7
    else assign 8
```

After annotation we found that the distribution of labels is extremely skewed. For that reason, we merged classes 2 and 3, and classes 6 and 7, respectively. Moreover, we removed instances of classes 4 and 8 from the corpus. The raw and clean counts of the classes in the dialogue corpus are distributed as follows.

Label	1	2	3	4	5	6	7	8
Raw	530	2	2209	3	454	23	76	2
Clean	530	2211		0	454	99		0

4 Representations and Dialogue Management

Before we discuss our choice of kernel functions, we detail the function f we would like to learn. The representations to be classified need to capture two aspects of the dialogue state: (1) how specific is the information gathered so far, and (2) how well does the highest ranking document from the current n-best list answer the users' question? The first aspect is expressed by the context representation, the second by the relationship between the gathered information and the highest-ranking document in the n-best list. This is the motivation behind modeling the dialogue manager as a classifier $a = f(c, l)$ in which the dialogue context c (including the last utterance of the user) and the document n-best list l supplied by the information retrieval engine determines the action a to be taken by the dialogue manager.

In the following sections, we describe how features for the representation of context c and the answer list l are extracted.

W: こちらは旅行ガイドです
This is the travel guide.
U: 京、京都で１日ゆったりとお寺巡りをしようと思っているんですが、京都駅から、お寺を、
お寺巡りをする、効率的というか、こうしたらいいんじゃないかっていうコースはありますか
I was thinking to take a small trip to Kyoto to see some temples, starting
from Kyoto station, is there something like an efficient course to do this?
W: はい、それでは京都駅周辺でお調べ致します
京都駅周辺をゆったりと回る、でよろしいですか
Yes then I will look for something around Kyoto station.
A quiet trip around Kyoto station, would that be okay?
U: はい
Yes.
W: それではJR京都駅から東寺、梅小路蒸気機関車館、門屋もてなしの文化美術館、西本願寺、
コウセイジ、京都タワー、そしてJR京都駅に戻ってくるというコースはいかがでしょうか
Then starting from Kyoto station, going to the Touji temple, to a steam train museum,
Kadoya museum, Nishi-Hongan temple, Kousei temple, Kyoto tower and back to Kyoto station,
would that be okay?

Figure 3: An extract from the dialogue corpus used. The letter *'U'* identifies user utterances, the letter *'W'* identifies wizard utterances.

4.1 Context Representation

We choose a simple discourse representation in which the user turns as well as information extracted from them are stored. More specifically, we memorize the user turn ut_T at time T, the set of keywords $kw(ut_T)$ extracted from ut_T by the question analysis module and the question type $qt(ut_T)$ determined by the question classifier. At time T, we also have access to an n-best list of newspaper articles $A_T = \langle a_T^1, \ldots, a_T^n \rangle$ which are returned from the question answering system.

The dialogue state at time T is then given by the three lists $\langle ut_1, \ldots, ut_t \rangle$, $\langle kw(ut_1), \ldots, kw(ut_T) \rangle$ and $\langle A_1, \ldots, A_T \rangle$. A dialogue state contains too much information to be used directly for classification purposes. We discuss several ways of extracting features relevant for classification in the following sections.

User input and context in spoken dialogue systems is often represented as fillers of a predefined set of slots. A straightforward adaptation to interactive restricted domain question answering systems is not straightforward, because it is difficult to determine the set of slots necessary for the application at hand. Since we intend to use the representations for classification purpose, we propose a *bag-of-word* approach that is often used in kernel-based methods.

4.1.1 Pure Context

While in task oriented dialogue systems, the representation of context is an accumulation of slot-filler part of some form, we cannot resort to this technique, because a predefined set of slots is not given. Therefore, we propose to generalize slot-filler representations such that the *information relevant to the information retrieval process* is memorized. We arrive at a bag-of-word representation of a sentence by tokenizing and tagging the sentence (Japanese does not use space between the words). Then, all content words are added to the bag-of-words. However, the chosen content words may be too specific or too vague for classification purposes. For that reason, we add all hypoernyms of all present nouns to the set.

In other words, our first proposal for context representation at time T is given by

$$C_T = Cl\left(\bigcup_{t=1}^{T} kw(ut_t)\right) \tag{1}$$

where Cl is the closure function mapping a set of words to a larger set that also contains all hypernyms. Informally, this is the set of all key words extracted from the user utterances up until now, augmented by the closure of concepts to include hypernyms.

4.1.2 Filtered through results

In information retrieval systems using large corpora co-occurence between terms is used as an additional source of information. In other words, the fact that two terms appear in the same documents more often than chance is used to address problems such as synonyms. When representing the context for classification, we would like to take advantage of this concept as well. We can do this in an indirect way by extracting relevant terms from the retrieved documents and add them to our context representation. This can be done as follows. Let N be the set of all nouns occurring in the dialogue training corpus. Furthermore, the document a_T^1 refers to the highest ranking newspaper article retrieved at time

T. An approximation of those nouns in the document relevant to the travel domain can be given by the intersection of the set of all nouns in the document with the set N. This leads to the following equation.

$$C_T = Cl\left(\bigcup_{t=1}^{T} kw(ut_t) \cap N \cap Cl(a_T^1)\right) \quad (2)$$

4.2 Answer Representation

In addition to a representation of the context, it is possible to represent the degree to which the retrieved documents answer the question asked by the user. We assume that this feature can be expressed by a relationship between the representation of the context and the retrieved documents.

4.2.1 Content Word Intersection

A straightforward approach to extract a relationship between the informational content in the context and in the retrieved document is to determine bag-of-words representations for each and calculate the intersection:

$$A_T = Cl\left(\bigcup_{t} kw(ut_t) \cap answer(a_T^1)\right) \quad (3)$$

where *answer* is a function that takes a document and extracts the sentence that (according to the question answering system) constitutes the answer to the users' question.

4.2.2 Alignment

A potential drawback to the bag-of-words representation described in section 4.2.1 is the loss of information that occurs as sentence structure is ignored. For this reason, our second approach is to determine an alignment between the user input sentence and the answer sentence extracted from the document. Figure 6 illustrates this idea by means of a toy example. This approach is motivated by the work by Brill et al (2002) described above. The detailed algorithm that determines the alignment is described in Appendix A.

In contrast to the features extracted in sections 4.1.1, 4.1.2 and 4.2.1, we cannot use a standard kernel to classify the extracted structure. Instead, we use the Hierarchical Directed Acyclic Graph kernel described in section 2.3.2 for classification purposes.

5 Dialogue Management

After having discussed the choice of representations, we proceed to describe dialogue management based on the representations. We recall from the introduction that our approach to dialogue management is to classify the dialogue state such that the resulting class is an indicator of the kind of action to be taken. The responsibility for carrying out the chosen action lies entirely with the generation component. Generation is an instance-based approach which takes sentences from the corpus and adapts them to the current dialogue situation as necessary. Examples for dialogues between inexperienced users and the system are shown in appendix A.

5.1 Classifiers and Kernels

For our experiments with binary classifiers, we use TinySVM as classifier, which is an implementation of Platt's fast SMO optimization algorithm. To choose the winning class, we use *one-vs-all* classification. For our experiments using multiclass classification, we use an implementation of the classifier described in (Weston and Watkins, 1999).

5.2 Instance-Based Generation

Given a query q asked by the user, and an answer n-best list $\langle a_1, \ldots, a_n \rangle$ generated by the question answering system, the classifier determines for each alignment $A(q, a_i)$ whether a_i is an answer candidate for q, and if so, the answer is added to the content for the information seeking question.

Another important question, not discussed in this paper, is how to turn the action a as determined by the classifier, into an acceptable utterance. Currently, we are pursuing an approach of instance-based generation, where appropriate sentences are selected from the dialogue corpus, adapted to the current context where necessary, and presented to the user. Details on the instance-based approach can be found in (Denecke and Tsukada, 2005). In short, the instance-based method for information seeking question generation works as follows. A dialogue with "similar context" is retrieved from the example corpus, the information-seeking question following the "similar context" is chosen, the content words in this question are determined and exchanged with content words from the current corpus, and the question is presented to the user.

5.3 Dialogue Management Algorithm

The dialogue management algorithm can be summarized as follows.

Input User query
Output Clarification question or answer

1 Perform standard question classification and keyword extraction on user query
2 Add extracted features to dialogue context
Add hypernyms to dialogue context
3 Perform standard IR using result of 1)

Add retrieved documents to dialogue context

4 Classify dialogue context to obtain action type

5 Retrieve user utterance from dialogue corpus similar to current situation

6 Adapt wizard utterance directly following user utterance to current situation

Answers	Context: Equ (1)	Context: Equ (2)
One-vs-all Equ (3)	76.3%	75.3%
Multiclass (Equ (3)	84.2%	80.4%
Alignment	62.2%	61.4%

Table 2: Accuracy of the classifiers

6 Evaluation

We combine the proposed representations proposed in sections 4.1.1, 4.1.2, 4.2.1 and 4.2.2 to obtain four different representations. In order to combine the features for context and answers, we distinguish whether the second kernel is a polynomial kernel (section 4.2.1) or a graph kernel (section 4.2.2). In the first case, we join the features for context and answers, taking care that the feature identifiers are distinct. In the second case, we create create a new graph node in the hierarchical graph representation and add all context features in this node.

6.1 The Question Anwering System

For our experiments, we used the Japanese question answering system SAIQA (Sasaki, 2002) as the building block. The text resources the system accesses are the complete documents of the *Mainichi Shinbun* dating from 1991 to 2002.

6.2 Methodology

We used a training set consisting of 151 dialogues to train the four different versions of Support Vector Machines. This is a 4 class classification problem; we used the cleaned version of the corpus. The Support Vector Machines were then used to predict the labels of the unseen corpus. In each case, we trained 4 classifiers, each singling out one of the four classes (one-against-all). The parameter C was set 1 in all experiments.

It should be noted that the purpose of this experiment is to determine an appropriate classifier for dialogue management to be used in a user study.

6.3 Results

The results of the four classifiers are shown in table 2. It can be shown that the context feature extraction according to equation (2) combined with the answer feature extraction according to equation (3) works best. We believe the structured alignment does not work better because the alignment is between one user utterance and one sentence in the retrieved document. In order to work well, more context should have been taken into account.

We also tried radial and neural kernels instead of the polynomial kernel, but found problems with both of them. The radial basis kernels showed excellent accuracy and precision, but had recall below 30 % for underrepresented classes. The training of SVMs with neural kernels would not converge, even when setting the parameter C to unusually large values.

In initial experiments, we tried to classify the structures resulting from the alignments as shown in figures 6 and 7. While the performance on the training set was around 90 percent, performance on previously unseen data was not much better than chance. This suggests that generalization could not take place. We hypothesize that this is due to the fact that the number of nodes and edges in the linguistic structures of question and answer outweighs the number of nodes and edges of the alignment. In other words, we suspect that overfitting occurs, and therefore the classifier does not generalize well. One way to overcome this problem is to remove some of the information from the alignment structures. We do this by removing all word related information, such as part of speech tag and semantic information. Yet, the simpler bag-of-word (or bag-of-concept, more appropriately)approach still outperforms the structured approach.

7 Discussion and Future Work

In this paper, we presented dialogue management techniques for interactive restrictive domain question answering systems. We proposed to learn classifiers for dialogue management based on representations extracted from dialogue context and retrieved documents from the question answering machines.

The main problem when going from task-oriented dialogue systems to interactive restricted-domain question answering systems is that the lack of task structure prohibits making simplifying assumptions as in task-oriented dialogue systems. In order to address this issue, we proposed a solution that consists of two parts: First, we use representations based on keywords extracted from the user utterances and concepts extracted from an ontology. These representations can be seen as a generalization of the slot/filler representations Future work includes evaluation with new users as opposed to trying to predict the actions in an existing dialogue corpus. Second, we use machine learning to learn the dialogue man-

agement function. We are currently preparing an evaluation with users to test the dialogue classification under realistic circumstances.

Acknowledgements

We would like to thank Takuya Suzuki for implementing parts of the instance-based generation system. Our thanks go to Hideki Isozaki and the members of the Knowledge Processing Group for discussions and support.

References

M. Asahara and Y. Matsumoto. 2000. Extended Models and Tools for High-Performance Part-of-Speech Tagger. In *Proceedings of The 18th International Conference on Computational Linguistics, Coling 2000, Saarbrücken, Germany.*

M. De Boni and S. Manandhar. 2003. An analysis of clarification dialogue for question answering. In *Proceedings of HLT-NAACL 2003, Edmonton, Canada.*

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2002. Data-Intensive Question Answering.

M. Collins and N. Duffy. 2001. Convolution Kernels in Natural Language Processing.

M. Denecke and H. Tsukada. 2005. Instance-Based Generation for Interactive Restricted-Domain Question-Answering Systems. In *Proceedings of the 2nd International Joint Conference on Natual Language Processing.* Springer Verlag.

M. Denecke and A.H. Waibel. 1997. Dialogue Strategies Guiding Users to their Communicative Goals. In *Proceedings of Eurospeech, Rhodos, Greece.* Available at http://www.opendialog.org/info_papers.html.

A. Ferrieux and M.D.Sadek. 1994. An Efficient Data-Driven Model for Cooperative Spoken Dialogue. In *Proceedings of the International Conference on Spoken Language Processing, Yokohama, Japan*, pages 979 – 982.

A. Flycht-Eriksson and A. Jönsson. 2003. Some empirical Findings on Dialogue Management and Domain Ontologies in Dialogue Systems - Implications from an Evaluation of Birdquest. In *Proceedings of the 4th* SigDial *Workshop on Discourse and Dialogue, Sapporo.*

D. Haussler. 1999. Convolution kernels on discrete structures. Technical report, UC Santa Cruz.

C. Hori, T. Hori, H. Tsukada, H. Isozaki, Y. Sasaki, and E. Maeda. 2003. Spoken interactive odqa system: Spiqa. In *Proc. of the 41th Annual Meeting of Association for Computational Linguistics (ACL-2003), Sapporo, Japan.*

S. Ikehara, M. Miyazaki, S. Shirai, A. Yokoo, H. Nakaiwa, K. Ogura, Y. Oyama, and Y. Hayashi. 1997. *The Semantic Attribute System, Goi-Taikei ? A Japanese Lexicon, Volume 1 (in Japanese).* Iwanami Publishing.

K. Kiyota, S. Kurohashi, and F. Kido. 2002. "dialog navigator": A question answering system based on large text knowledge base. In *Proceedings of The 19th International Conference on Computational Linguistics, Coling 2002,Taipei, Taiwan.*

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*, pages 63–69.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A Stochastic Model of Human Machine Interaction for Learning Dialog Strategies. *IEEE Transactions on Speech and Audio Processing*, 8:11–23.

M. Minock. 2005. Where are the 'Killer Applications' of Restricted Domain Question Answering? In *Workshop on Knowledge and Reasoning for Answering Questions, Edinburgh.*

T. Misu and T. Kawahara. 2005. Dialogue Strategy to Clarify Users Queries for Document Retrieval System with Speech Interface. In *Proceedings of Interspeech 2005 (Eurospeech), Lisbon.*

A. Rudnicky and X. Wu. 1999. An agenda-based Dialog Management Architecture for Spoken Language Systems. In *Proceedings of the Workshop on Automatic Speech Recognition and Understanding.*

Y. Sasaki. 2002. Ntt's qa systems for ntcir qac-1. In *Working Notes of the Third NTCIR Workshop Meeting*, pages 63–70.

S. Small and T. Strzalkowski. 2004. Hitiqa: Towards analytical question answering. In *Proceedings of The 20th International Conference on Computational Linguistics, Coling 2004,Geneva Switzerland.*

J. Suzuki, T. Hirao, Y. Sasaki, and E. Maeda. 2003. Hierarchical directed acyclic graph kernel: Methods for structured natural language data. In *Proc. of the 41th Annual Meeting of Association for Computational Linguistics (ACL-2003), Sapporo, Japan*, pages 32–39.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory.* Springer.

J. Weston and C. Watkins. 1999. Support vector machines for multi-class pattern recognition. In *Proceedings ofthe Seventh European Symposium on Artificial Neural Networks.*

Robert Wilensky, Yigal Arens, and David Chin. 1984. Talking to unix in english: an overview of uc. *Communications of the ACM*, 27(6):574–593.

A An Example Dialogue

The following example dialogue transcript was obtained during a user study with a user that had not used the system before.

U : 横浜の観光施設を教えてください
: Please tell me about sightseeing attractions in Yokohama.
S : どういった観光施設をお期待するんですか
: What kind of sightseeing attractions are you looking for?
U : 家族で楽しめるところを
: Something to enjoy with my family.
S : それではシルク市場や、横浜開港資料館はいかがでしょうか
: In that case, how about silk market or Yokohama Port Museum?
U : 横浜開港資料館はどんなところですか
: What is Yokohama Port Museum?
S : じゃ横浜明治の冊にはけっこういろいろあるって訳ですよね
: Then you mean that Yokohama Meiji has many <unintelligible>?
U : 横浜にはどんなイベントがありますか
: What kind of events are currently taking place in Yokohama?
S : 乗船会
: Meeting for sailboat fans.

Figure 4: Example dialogue. The words replaced by the instance-based generation are highlighted in grey.

B Alignment Algorithm

The purpose of our work is to build a classifier that determines whether the sentence shown in figure 5 (b) is an answer to the query shown in figure 5 (a). In other words, we are interested in classifying the relation between sentence 5 (a) to sentence 5 (b). Therefore, we need to add structure to 5 (a),(b) that makes explicit the relationship between the dependency structures. We do this by aligning the dependency trees for the question and the answer with each other. Then, we form a graph consisting of the dependency tree of the question, the dependency tree of the answer and the edges representing the alignment. In the following, we refer to the surface form of the question and answer as q and a, respectively, their dependency trees as $t(q)$ and $t(a)$, and a possible alignment of $t(q)$ with $t(a)$ as $A(q, a)$.

An example of one possible alignment for the structure resulting from the alignment of the question and answer shown in figure 5 is shown in figure 6. The resulting structure is classified using the Hierarchical Directed Acyclic Graph Kernel; counting as a positive example if the presented answer is indeed the correct answer to the question and as a negative example if not. In the following, we present the preprocessing before the alignment and propose two different ways to perform the alignment.

B.1 Preprocessing

Before the alignment can take place, we need to perform some preprocessing in order to address the differences between question and answer. We are primarily concerned with two issues. First, we need to address the fact that nodes containing the question word (such as *who, what, where* and so on) need to be treated differently. We do this as follows. From the training corpus, we associate with each question type the used question words. Then, in order to identify the node containing the question word in the dependency tree of the question, we determine the nodes that are associated with a word in the list.

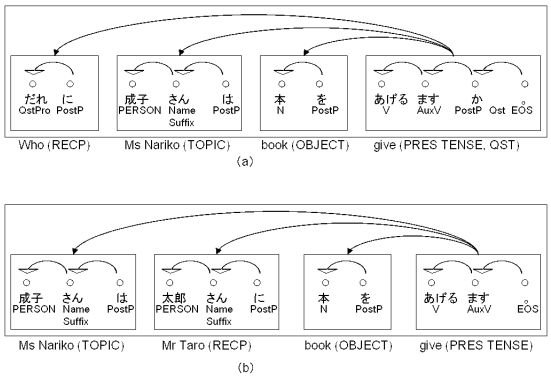

Figure 5: Analysis

B.2 Detailed Alignment

The detailed alignment is an attempt at aligning as many nodes as possible in $t(q)$ with nodes in $t(a)$. In order to abstract away the particular realization of question and answer, we do not consider nodes whose part of speech tag is not a noun, verb, adjective or adverb. In the following we refer to a node from the question or answer dependency tree as n_q, and n_a, respectively. The i-th attribute of a node n is indicated by $attr_n(i)$. The part of speech of n is notated as $pos(n)$. The i node in contained in n can be accessed by $node(n, i)$. We distinguish between word and chunk nodes. A word node is a node containing a word appearing in the surface form of the question or answer, but does not contain other words. Examples for word nodes include the nodes labeled with the words 'Nariko' or 'book' in figure 5. A chunk node is a node containing other nodes. Chunk nodes are shown as square boxes in figure 5. We are interested in determining the value $V(n_q, n_a)$ of aligning node n_q with node n_a. We distinguish three cases: the value of the alignment if both nodes are word nodes, the value of the alignment if both nodes are chunk nodes, and the remaining case. A value of $V(n_q, n_a) = -1$ represents the fact that n_q and n_a are not aligned.

B.2.1 Value of Aligning two Word Nodes

We first consider in which both n_q and n_a contain the representation of a word. The nodes carry information on the surface string, the uninflected word, the part of speech, and semantic tag. The value of the alignment between two word nodes is given by the number of attributes contained in the question node that are also contained in the answer node. This characteristic is intended to capture the fact that information present in the answer but not in the question does not hurt the association, while the other round it does.

$$V(n_q, n_a) = \begin{cases} |\{i : attr_{n_q}(i) = attr_{n_a}(i)\}| & \text{if } pos(n_q) \in list \\ -1 & \text{otherwise} \end{cases}$$

B.2.2 Value of Aligning two Chunk Nodes

A chunk node contains references to word nodes, and, since the graphs are hierarchical, other chunk nodes. For alignment purposes, we consider a chunk node a set of nodes. We do not take those word nodes into account. Given two chunk nodes, the cost of associating between them is calculated recursively according to

$$V(n_q, n_a) = \max_{\pi \in \Pi(s)} \sum_j V(node(n_q, j), node(n_a, \pi(j)))$$

where $\Pi(s)$ is the set of all permutations of $\{1, \ldots, s\}$ and s is the number of nodes contained in n_q.

The value $V(n_q, n_a)$ equals 0 if one node is a word node while the other is a chunk node.

Dynamic programming is used to determine a minimal association between the structures of the query and the potential. A solution to the example sentences from figure 5 is shown in figure 6. Once the alignment has been determined, the edges $\langle n_q, n_a \rangle$ for which $V(n_q, n_a) > 0$ are added to the graph. In addition, each node n in $t(q)$ that is not aligned with some node in $t(a)$, we add a new node n' and align n with n'. The resulting structure is shown in figure 6.

B.3 Sloppy Alignment

Sloppy alignment is identical to detailed alignment except that after alignment only those edges $\langle n_q, n_a \rangle$ for which $V(n_q, n_a) > 0$) such that n_q is a chunk node are added to the graph. Figure 7 shows the sloppy alignment.

B.4 Thinning

In order to avoid memorization, we remove features from the graph structures. We propose to thin out the graph and make the alignment structure more visible. The resulting graph of the example is shown in figure 8.

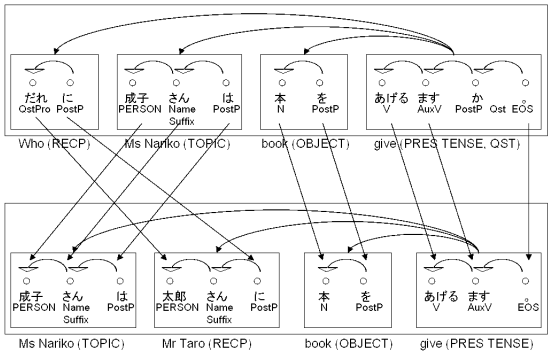

Figure 6: One possible alignment of the dependency trees shown in figure 5. The alignment of the chunk nodes is not shown for clarity, but is added to the graph as well.

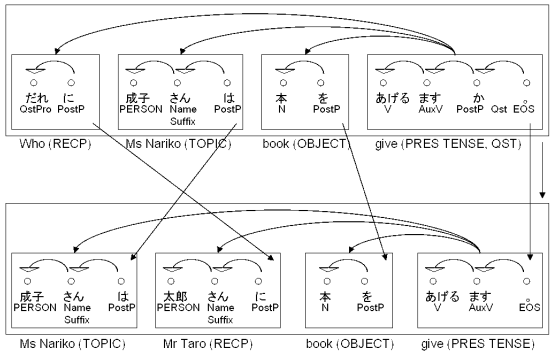

Figure 7: One possible alignment of the dependency trees shown in figure 5.

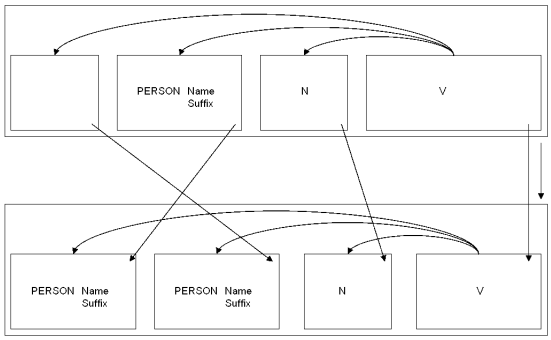

Figure 8: Alignment shown in figure 7 after thinning.