Instance-Based Generation for Interactive Restricted Domain Question Answering Systems

Matthias Denecke and Hajime Tsukada

NTT Communication Science Laboratories,
2-4 Hikaridai, Seika-Cho, Soraku-gun, Kyoto
{denecke, tsukada}@cslab.kecl.ntt.co.jp

Abstract. One important component of interactive systems is the generation component. While template-based generation is appropriate in many cases (for example, task oriented spoken dialogue systems), interactive question answering systems require a more sophisticated approach. In this paper, we propose and compare two example-based methods for generation of information seeking questions.

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.

Spoken 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. 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. In this paper, we are interested in generation of information seeking questions in interactive question-answering systems.

1.1 Our System

We implemented a system that combines features of question answering systems with those of spoken dialogue systems. We integrated the following two features in an interactive restricted domain question answering 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.

1.2 Problem Addressed in This Paper

Restricted domain question answering systems can be deployed in interactive problem solving solutions, for example, software trouble shooting. 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 [5], 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 specifically address the problem of how to generate information seeking questions in the case of ambiguous, vague or ill-defined user questions. We assume that the decision of whether an information seeking question is needed is made outside of the module described here. More formally, the problem we address can be described as follows:

Given   1. A representation of the previous interaction history, consisting of user and system utterances, and retrieval results from the IR subsystem,
          2. A decision for a information seeking question
Produce   An information seeking question.

Problems of this kind have appeared traditionally in task oriented spoken dialogue systems, where missing information needs to be prompted. However, in the case of spoken dialogue systems, question generation is typically not a substantial problem: the fact that the back-end is well-structured allows for simple template-based generation in many cases. For example, missing values for database queries or remote method invocations can be queried that way. (But see also Oh and Rudnicky [7] or Walker *et al* [12] for more elaborated approaches to generation for spoken dialogue systems).

In our case, however, a template-based approach is unrealistic. This is due to the unstructured back-end application. Unlike as spoken dialogue systems, we cannot make assumptions over what kind of questions to ask as this is determined by the result set of articles as returned by the information retrieval engine. Existing interactive question-answering systems (see section 7.1 for a more detailed description) either use canned text on dialogue cards [5], break down the dialogue representation into frames and then techniques from spoken dialogue systems [8], or make simplifying assumptions to the extent that generation essentially becomes equivalent to template-based generation.

1.3   Proposed Solution

For reasons discussed above, we propose an example-based approach to generation. More specifically, we use an existing dialogue corpus to retrieve appropriate questions and modify in order to fit the situation at hand. We describe two algorithms for instance-based natural language questions generation by first selecting appropriate candidates from the corpus, then modifying the candidates to fit the situation at hand, and finally re-rank the candidates. This is an example of a memory-based learning approach, which in turn is a kind of a case-based reasoning. To the best of our knowledge, this is the first work addressing the problem of example-based generation information seeking questions in the absence of a structured back-end application.

2   Instance Based Natural Language Generation

In this section, we review the background in memory-based learning and its application in natural language generation.

2.1 Memory-Based Reasoning

Memory-based reasoning (MBR) is often considered a subtype of *Case-based reasoning*. Case-based reasoning was proposed in the 80's as an alternative to rule-based approaches. Instead of expressing regularities about the domain to be modeled in rules, the primary knowledge source in case-based reasoning is a memory of cases representing episodes of encountered problems. Generating a solution to a given problem consists of retrieving an appropriate case from memory and adapting it to the problem at hand.

MBR solves problems by retrieving stored precedents as a starting point for new problem-solving (e.g., [9]). However, its primary focus is on the retrieval process, and in particular on the use of parallel retrieval schemes to enable retrieval without conventional index selection. One aspect of memory-based systems is to choose a distance that appropriately selects candidate exemplars.

Memory-based reasoning has been applied to machine translation, parsing, unit selection text-to-speech synthesis, part-of-speech tagging, and others. An overview of memory-based approaches to natural language processing can be found in the introduction to the special issue [2].

2.2 Statistical and Instance-Based Generation

The most prominent example for statistical generation is NITROGEN [6]. This system has been designed to allows large scale generation while requiring only a minimal knowledge base. An abstract meaning representation is turned into a lattice of surface sentences using a simple keyword based grammar. Using statistical information acquired from a corpus, the sentences in the lattices are re-ranked to determine the optimal surface string.

More recently, example-based natural language generation using a corpus was proposed [11]. It is assumed in this work that content determination has already taken place and the input has been broken down to sentence-size pieces. The approach is to use a learned grammar to generate a list of candidates using a traditional chart based generation algorithm. The grammar is learned using statistical methods. During generation, edges that are added to the chart are ranked depending on their distance to the closest instance in the example base. This is where the memory-based approach comes into play. In order to allow for careful generalization in the instance base, the authors propose to add a list of tag ("slots") with which the corpus is annotated. Based on this annotated corpus, a semantic grammar is learned. For ranking the edge based on the instances, the authors propose the well-known *tf-idf* scheme with the difference that those words that are annotated with a semantic tag are replaced by their tag.

3 Kernels

Memory-based learning requires a distance metric in order to identify instances similar to the problem at hand. We propose to use *convolution kernels* as distance metric. A kernel K can be seen as a generalized form of a distance metric that performs the following calculation

$$K(x, y) = \langle \phi(x), \phi(y) \rangle,$$

where ϕ is a non-linear mapping from the input space into some higher dimensional feature space, and $\langle \cdot, \cdot \rangle$ is the inner product in the feature space. Calculating the inner product in some space of higher dimension than the input space is desirable for classifiers because non linearly separable sets can be linearly separated in the higher dimensional feature space. Kernel methods are computationally attractive because the kernel can calculate the mapping and the inner product implicitly rather than explicitly determining the image under ϕ of the input.

While *Bag-of-Words* techniques can be employed as an approximation to derive feature vectors for classifiers, the loss of structure is not desirable. To address this problem, Haussler [3] proposed *Convolution Kernels* that are capable of processing structured objects x and y. The structured objects x and y consist 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 objects of various kinds, and results have been reported for string kernels and tree kernels.

3.1   Hierarchical Tree Kernel

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:

$$K(x, y) = \sum_{\mathbf{x} \in R^{-1}(x)} \sum_{\mathbf{y} \in R^{-1}(y)} \prod_1^D K_d(x_d, y_d)$$

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.

Suzuki et al [10] 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.

4   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. Altogether, 20 users participated in the

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?

Fig. 1. Extract from the dialogue corpus

data collection effort. Each user contributed to 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 1 shows the first part of a dialogue from the corpus.

5   Generation Algorithm

5.1   Overview of the Algorithm

We now describe our algorithm informally. Given the dialogue history up until now, the last user utterance and the result list as a response to the last user utterance, it is the task of the algorithm to generate an appropriate question to elicit more information from the user. Recall an external dialogue module (not described in this paper) decides whether an information seeking question should be generated (as opposed to, say, turning the information found in the highest ranking article into an answer).

Informally, the algorithm works as follows. Initially, the dialogue corpus is preprocessed, including word segmentation and part-of-speech labeling (see section 5.2). In step 1, a ranked list of question candidates is generated (see section 5.3). In step 2, for each of the candidates, a list of change positions is determined (see section 5.4). These indicate the part of the questions that need to be adapted to the current situation. Subsequently, the portions indicated by the change positions are replaced by appropriate constituents. In the step 3, the candidates generated in the previous step are re-ranked (see section 5.5). Re-ranking takes place by using the same distance as the one in step 1. The highest ranking candidate is then presented to the user.

5.2   Corpus Preprocessing

Since Japanese does not provide word segmentation, we need to preprocess the corpus. The corpus consists of a set of dialogues. Each dialogue consists of a set of utterances. Each utterance is annotated for speaker and utterance type. In a dialogue, wizard and user utterance strictly alternate, with no interjections.

Preprocessing is done as follows. Each utterance is stripped of its annotations and presented to the part-of-speech tagger *Chasen* [1]. Chasen segments the input sentence, reduces inflected words to their base forms and assigns part of speech tags to the base forms. We use the notation $cw(u)$ to designate the content words in utterance, sentence or newspaper article u. For our purposes, content words are adjectives, nouns and verbs, de-inflected to their base form, if necessary. A subsequent processing step assigns semantic labels and named entity classes to the de-inflected word forms.

5.3   Sentence Selection

In order to understand the motivation for our approaches to sentence selection, it is necessary to recall the context in which sentences are selected. We would like to find a information seeking question similar to the one we want to generate. The question to be generated is determined by the dialogue context. A natural approach is to choose a bag-of-word distance measure for sentences, define a distance for partial dialogues based on this distance and then choose the dialogue, and a sentence from that dialogue with the lowest distance.

It turns out, however, that this approach does not work too well. One problem is that in the beginning of a dialogue not many informative words are contained in the utterances, therefore making an informed selection of utterances difficult. The point of this paper is to determine how to overcome this problem. In the following two sections, we propose two approaches. The first uses additional information in the retrieved documents, and the second uses additional syntactic and semantic information when calculating the distance between sentences. Both methods consists of calculating a score for candidate sentences and selecting the highest ranking one.

Method 1. Information retrieval over large corpora works well due to the redundancy in the document data, a fact that for example Latent Semantic Indexing exploits. The principal idea of the first method is to use the redundancy in the unrestricted document corpus when scoring sentence candidates. Instead of determining the bag-of-word score between a candidate sentence and the query sentence, we submit the information extracted from the candidate dialogue and the current dialogue to the information retrieval engine, resulting in two n best lists of articles L and L'. In order to score the degree of similarity, we determine the the intersection of content words in the retrieved articles. The larger the intersection, the higher the score is to be ranked. In order to take relevance in the result set into account, the scores are discounted by the position of the article in the n best list. More specifically, we calculate the similarity score between the current dialogue and an example dialogue as follows. Let d be the currently developing dialogue consisting of t user utterances and $u_1, \ldots u_t$ be the user utterances in the current dialogue up until now. Furthermore, let d' be an example dialogue from the corpus and let $u'_1, \ldots u'_{t'}$ be the first t' user utterances in the example dialogue. Then:

1. Form the union of content words $CW = \bigcup_t cw(u_t)$, $CW' = \bigcup_{t'} cw(u'_{t'})$
2. Submit two queries to the information retrieval engine consisting of CW and CW', respectively and obtain two article n best lists L and L'.
3. Calculate the similarity score according to

$$sim(u_t, u'_{t'}) = \sum_{l \in L} \sum_{l' \in L'} \frac{cw(l) \cap cw(l')}{rank(l) + rank(l')}$$

Method 2. In the first method described above, we seek to overcome poor scoring function by adding redundancy from the information retrieval engine. The second method we propose attempts to improve scoring by adding syntactic and semantic structure to the distance metric. More specifically, we directly compare the last user utterance in the current dialogue with the last utterance in the example dialogue, but do so in a more detailed manner. To this end, we determine the similarity score as the output of the hierarchical directed acyclic graph kernel. The similarity is thus defined as $sim(u_t, u'_{t'}) = K(u_t, u'_{t'})$.

5.4   Sentence Adaptation

The adaptation of the highest ranking question to the current dialogue consists of four steps. First, we determine the location(s) where change should take place. Second, we determine constraints for the substituting constituent. Third, we determine a list of substituents for each location of change. Fourth, we replace the phrase(s) at the location(s) of change with the highest ranking element from the corresponding list of substituents.

Determining Locations of Change. After the example sentences have been retrieved from the corpus, we need to determine where and how the questions need to be adapted to the current dialogue. We determine the locations of change l_i by identifying suitable head words of phrase to be exchanged. What are the criteria for suitable head words? Recall that the example sentences are drawn from dialogue similar in topics but in which the content words are exchanged. This limits the part-of-speech of the words to be exchanged to nouns and verbs. Therefore, we construct a list l of nouns and verbs that are part of the retrieved sentence but cannot be found in the current user query. Second, since we are interested in replacing those content words that are specific to the retrieved dialogue with those specific to the current dialogue, we would like to incorporate some measure of informativeness. For that reason, we determine the unigram count for all content words in l. High ranking candidates for change are those words that are specific (i.e., have a low unigram count above a certain threshold).

Constraints for Substituents. The constraints for the substituents are given by the semantic and syntactic information of the phrase at the change location. More specifically, the constraints include the following features: Part of speech, type of named entity, if applicable (the type includes location, state, person name and so on), and semantic class.

Determining Substituents. After having determined the change locations and constraints of the substituents, we proceed to determine the substituents.

The primary source for substituents are the retrieved newspaper articles. However, since we wish to apply the generation component in a dialogue system, we need to take implicit confirmation into account as well. For this reason, we determine whether a phrase matching the phrase at change location l_i occurs before l_i previously in the dialogue. If this is the case, the source for the substituent is to be the current dialogue.

Given the constraints for a change location determined in the previous step, we add all content words from the highest ranking article to the candidate list for that change location. The score for a content word is given by the number of constraints it fulfills. Ties are broken by unigram counts so that rare words get a higher score due to their informativeness.

Application of Change. Applying the change simply consists of removing the phrase whose head word is located at the change location and replacing it with the highest ranking word from the candidate list for that score.

5.5   Reranking

The previous steps produce a list of sentence candidates. For each of the sentence candidates, we calculate the similarity between the generated sentence with the sentences from a small corpus of desirable sentences. Finally, the sentence with the highest score is presented to the user. Examples of generated sentences are shown in figure 2. The complete algorithm is given in figure 3.

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?

Fig. 2. Generated questions. The substituent in the first question comes from the dialogue context, while the other substituents come from retrieved articles.

6   Evaluation

The evaluation was done as follows. We divided the corpus in a example base and a test set. The example base consists of 151 randomly selected dialogues, the test set consists of the remaining 50 dialogues. From each of the test examples, we supplied the initial wizard greeting and the initial user utterance as context for the dialogue. Given this context, each method generated an n best list consisting of 3 information seeking questions.

The generated lists were labeled by three annotators according to the following criteria. For each of the three questions in the n best lists, the annotators had to determine a syntactic, a semantic and an overall score. The scores range over the labels *poor, acceptable, good*. The same score could be assigned more

Input: Preprocessed dialogue corpus $C = \{d'_1, \ldots, d'_n\}$
Current dialogue d with user utterances $u_1, \ldots, u_t$

Output: Information seeking question

Step 1: Determine $sim(u_t, u'_{t'})$ for all user utterances $u'_{t'}$ from the dialogue corpus
Select the $w'_1, \ldots w'_k$ wizard utterances directly following the k
highest ranking utterances

Step 2: **for each** $w'_i \in \{w'_1, \ldots, w'_k\}$:
Determine change locations $l_1, \ldots, l_l$
for each $l_j \in \{l_1, \ldots, l_l\}$
Determine list of substituents $s^1_{ij}, \ldots, s^p_{ij}$

Generate modified sentence list $v_1, \ldots, v_m$ by replacing substituents
at change locations

Step 3: Determine and return highest ranking v_{i^*}.

Fig. 3. Generation algorithm

than once, for example, in case the sentence selection algorithm produced an unreliable candidate, the overall score for all three sentence candidates could be *bad*. Furthermore, the evaluators had to re-arrange the 3 best list according to the quality of the generated questions. Finally, the annotators had provide a sentence they consider good. For easy comparison, the symbolic scores *poor, acceptable, good* translate to 0,0.5 and 1, respectively, in the tables below.

6.1   Scoring Results

The results of the three best syntactic and semantic sentence scoring are shown in table 1 (a) and 1 (b). The inter-annotator agreement is given by their kappa scores for each method separately. Table 1 (c) shows the average of syntactic and semantic scores. The kappa coefficient for the inter-annotator agreement for these scores are 0.68, 0.72, and 0.71, respectively.

The syntactic scores rank higher than the semantic scores. This is explained by the fact that the corpus contains syntactically relatively well-formed example sentences, and the replacement operator, in addition to being constrained by

Table 1. Average of syntactic and semantic scores

	Method 1	Method 2		Method 1	Method 2		Method 1	Method 2
1	0.796	0.800	1	0.573	0.393	1	0.685	0.596
2	0.657	0.790	2	0.393	0.426	2	0.525	0.608
3	0.787	0.780	3	0.416	0.376	3	0.602	0.578
	(a)			(b)			(c)	

part-of-speech as well as semantic information, does not have much opportunity to create a syntactically malformed sentence. Furthermore, method 1 produces sentences that are semantically more accurate than method 2.

6.2   Ranking Results

In order to determine the quality of the ranking, the annotators had to rerank the generated questions. We determine the distance between two rankings according to the Edit distance. Since the generated lists are only of length 3, there are only three possibilties: the lists are equal (edit distance 0), one element in both lists is the same (edit distance 2), and no element in the lists is the same, (edit distance 3). In order to allow easy comparison with the table above, we award scores of 1, 0.5 and 0 for edit distances of 0, 2 and 3, respectively (i.e., 1 is best, 0 is worst). The annotators were asked to rank the questions according to syntactic criteria alone, semantic criteria alone and all criteria. The results are shown in Table 2.

Table 2. Comparison of ranking: Syntactic, semantic and overall

	Method 1	Method 2		Method 1	Method 2		Method 1	Method 2
1	0.493	0.893	1	0.720	0.873	1	0.766	0.853
2	0.813	0.860	2	0.760	0.780	2	0.740	0.726
3	0.767	0.227	3	0.567	0.353	3	0.573	0.213
	(a)			(b)			(c)	

It can be seen that method 2 ranks the example sentences in a way that is more in line with the choices of the annotators than method 1.

6.3   Quality of Ranking

We hypothesize that the differences in the performance of the algorithms is due to the different selection mechanisms. In order to validate this point, we asked the three annotators to each provide one utterance they would rank highest for each system question (called *gold standard*). Then, we formed a list of 6 sentences $u'_1, \ldots u'_6$ (3 generated by the generation algorithm and 3 by the annotators) and compared for each dialogue context the scores $sim(u_t, u'_i)$ for those 6 sentences where u_t is the user utterance from the corresponding test case. We expect a perfect ranking algorithm to value the gold standard as least as high as any sentence from the corpus, and to value the gold standard higher every time the annotators found the generated sentences faulty. It turns out that method 1 places the sentences of the gold standard in the top 3 in 42.3% of the cases while method 2 does this in 59.3% of the cases.

7   Discussion

It can be seen that in general, method 1 produces higher quality sentences while method 2 ranks the sentences better. We interpret this as follows. For sentence selection, the redundancy as provided by the IR engine is helpful, whereas for ranking of example sentences, the additional structure as expressed in the kernel helps.

7.1   Related Work

Kiyota and colleagues [5] describe an interactive restricted domain question answering system where users can interactively retrieve causes for problems with a computers' operating system. Here, the problem of missing structure is solved by providing so-called *dialogue cards* which provide the knowledge necessary for dialogue processing. A dialogue card contains keywords, a question as asked by the user in natural language (for example "Windows does not boot"), an information seeking question to be issued by the system (for example "Which version of Windows do you use") and a list of options associated with actions. The actions are executed in function of the users' answer to the question. Dialogue processing takes place by retrieving relevant dialogue cards, where relevance is determined by matching the users' question and keywords with the question and keywords noted on the dialogue card. Compared to our method, this method requires substantially more structure to be represented in the dialogue cards and is therefore more expensive to develop. Furthermore, the absence of any sort of change operators to adapt the question from the dialogue card to the current situation does not provide as much flexibility as our method. On the other hand, the highly structured dialogue cards give the developers more control (at the price of a higher development cost) over the systems behavior than our method and is therefore less risky in situations where failure is expensive.

In Small *et al* [8], retrieved documents are forced into frame structures. Mismatches or between the fillers of the frame structures or missing fillers trigger information seeking questions to the user. While the generation as it is actually used is not described in the paper, we believe that the frames provide sufficient structure for template-based approaches.

Hori and coworkers [4] developed an interactive question answering system based on a Japanese newspaper corpus. The purpose of information seeking questions is to prompt the user for missing or disambiguating information. From a generation point of view, strong assumptions are made on the surface form of the generated information seeking question. More specifically, ambiguous keywords are combined with disambiguating options by means of the Japanese particle 'no'.

7.2   Summary

To summarize, the presented approaches attempt in different ways to compensate for the lack of structure in an question answering system. Structure can be provided explicitly as in the case of the dialogue cards, can be introduced during processing as in the case of the frame-based document representations, and can be assumed in the target expression as in the case of the generation templates. In contrast to the described methods, our method does not require an explicit representation of structure. Rather, the structure is given by whatever structure the kernel and the change operators construct during generation. In other words, the structure our approach uses is (1) restricted to the question to be generated and does not apply to the document level, and (2) in tradition with the lazy learning characteristics of memory-based approaches is generated on the fly on an as-needed basis, as opposed to being dictated from the outset at design time.

Acknowledgements

References

1. 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.*
2. W. Daelemans. 1999. Introduction to the Special Issue on Memory-Based Language Processing. *Journal of Experimental and Theoretical Artificial Intelligence.*
3. D. Haussler. 1999. Convolution kernels on discrete structures. Technical report, UC Santa Cruz.
4. 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.*
5. 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.*
6. I. Langkilde and K. Knight. 1998. Generation that exploits Corpus-Based Statistical Knowledge. In *Proceedings of the Conference of the Association for Computational Linguistics (COLING/ACL).*
7. A.H. Oh and A. Rudnicky. 2000. Stochastic Language Generation for Spoken Dialogue Systems. In *ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 27–32.
8. 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.*
9. C. Stanfill and D. Waltz. 1986. Toward Memory-based Reasoning. Communications of the ACM, vol. 29, pages 1213-1228.
10. 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.
11. S. Varges and C. Mellish. 2001. Instance-based natural language generation. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1–8.
12. M. Walker, O. Rambow, and M. Rogati. 2001. SPoT: A Trainable Sentence Planner. In *Proceedings of the North American Meeting of the Association for Computational Linguistics.*