Natural Language Query Recommendation in Conversation Systems

Shimei Pan
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
shimei@us.ibm.com

James Shaw*
Ask.com
343 Thornall Street, 7th floor
Edison, NJ 08837
James.shaw@ask.com

Abstract

In this paper, we address a critical problem in conversation systems: limited input interpretation capabilities. When an interpretation error occurs, users often get stuck and cannot recover due to a lack of guidance from the system. To solve this problem, we present a hybrid natural language query recommendation framework that combines natural language generation with query retrieval. When receiving a problematic user query, our system dynamically recommends valid queries that are most relevant to the current user request so that the user can revise his request accordingly. Compared with existing methods, our approach offers two main contributions: first, improving query recommendation quality by combining query generation with query retrieval; second, adapting generated recommendations dynamically so that they are syntactically and lexically consistent with the original user input. Our evaluation results demonstrate the effectiveness of this approach.

1 Introduction

Mixed initiative conversation systems allow users to interact with computers using speech or natural language. However, these systems have not been used widely in practice mainly due to their limited input understanding capabilities. When interpretation errors occur, a user frequently gets stuck and cannot recover. Figure 1 shows an example in which the user cannot recover from repeated interpretation errors.

U1: Show the mean home price and household income of Pleasantville.
S1: Sorry, I don't understand you. Please rephrase your request.
(S1': Sorry, I don't understand. Do you mean "Show the ***average*** home price and household income of Pleasantville")
U2: Show the ***median house*** price and household income of Pleasantville.
S2: Sorry. I don't understand. Please rephrase your question.
U3 :Show the mean home price and ***the mean*** household income of Pleasantville
S3: Sorry, I still don't understand ....

Figure 1: An Example

This simple dialog demonstrates two typical problems in dialog systems: limited input understanding capability and a lack of proper guidance when errors occur. Limited input understanding capability is what got the user into trouble originally. In *U1*, the system was not able to understand "*mean*". Moreover, from *U2-U3*, without any guidance from the system, the user tried twice to recover from this problem without much success. Instead, if the system had responded with S1', it would be much easier for the user to rephrase successfully.

To enhance a user's experience and improve the robustness of human-computer interaction, most existing approaches focus on improving a system's interpretation capability directly. Nonetheless, such improvements may still be limited since they would never cover the entire range of user expressions for any non-trivial application. Alternatively, we propose to improve the robustness of system-user interaction by providing proper guidance on what the system can do so that users know how to adapt their queries to those within the system's capability. Figure 2 summarizes our approach. Assume U contains all the possible user expressions in an application and S has all the expressions a system understands. Since S is usually a subset of U, when a user query u1 is outside of S, it will cause interpretation errors. When this occurs, our system will provide proper guidance on what the system can do so that users know how to revise u1 to s1.

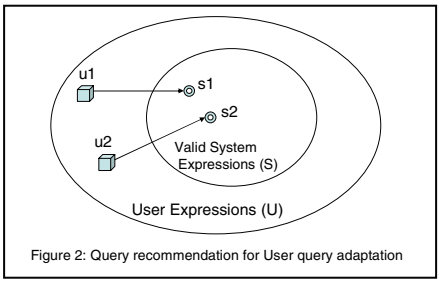

Figure 2: Query recommendation for User query adaptation

To make a system's interpretation capability apparent to a user in context, we propose a hybrid query recommendation framework in which we combine a retrieval-based approach with dynamic query generation. As a result, the hybrid system not only recommends queries based on the examples from a query corpus. It also dynamically composes new queries so that it can recommend queries that are close

* This work was conducted when the author was at IBM

to the user's original intent, which might be beyond the scope of pre-stored examples.

Our approach is embodied in an intelligent multimodal conversation system supporting four different applications including a real estate application that helps potential buyers to search for residential properties (Zhou et al., 2006). In this system, users may enter their queries using multiple input modalities such as speech, text, and GUI. In this work, we focus on queries typed in as text. To respond to a user's request, the system dynamically generates a multimedia presentation that includes automatically generated spoken utterances and graphics. When interpretation errors occur, the system dynamically generates several recommendations so that the user can select one of them to replace the original query or make further revisions if desired.

In the rest of the paper, after a brief discussion of related work, we explain the technical details of our approach. We then present our evaluation results.

2 Related Work

Previously, many approaches have been proposed to make a conversation system's capability apparent to users, such as tutorials or context-sensitive help. Tutorials (Kamm et al., 1998) are often too brief to cover all valid queries. Even if they manage to do so, users may not be able to remember and apply them to the current task. Alternatively, context-sensitive help systems (Hastie et al., 2002, Stein, 1998) present relevant examples in context. However, existing approaches for context-sensitive help have limitations. For example, finite state machine-based approaches (Walker et al, 1998) do not scale well. Depending on the granularity of the predicted classes, a decision tree-based help (Hastie et al., 2002) may be too coarse to provide guidance on the exact wording. Moreover, retrieval-based recommendation has been used in applications like spelling recommendations in Google spell check (Google, 2006). We have also used retrieval-based query recommendation in conversation applications (Pan et al., 2005). However, depending on the coverage of the examples in a system, the most relevant pre-stored example may not be close enough to be helpful. In contrast, the approach we propose here combines retrieval-based approach with dynamic query generation to provide scalable, fine-grained context-sensitive help based on the current user query.

Our work on dynamic recommendation generation also offers several contributions to the area of natural language generation. For example, traditionally, content selection is done either based on domain heuristics (McKeown et al., 1997) or content models learned from a corpus (Duboue and McKeown. 2003, Barzilay and Lapata, 2005). In contrast, our content selection is done based on the analysis of the current user query as well as the categorization of an interpretation error. For sentence generation, in addition to the grammaticality of generated sentences (Shaw. 1998, Langkilde. 2000, Walker et al., 2002) we also focus on using a cascade model to minimize the unnecessary difference between the original user query and system's recommendations.

In the following, we first explain the hybrid query recommendation approach, and then provide evaluation results.

3 Hybrid Query Recommendation

To achieve robust, scalable, fine-grained query recommendation, we extend our previous work on retrieval-based query recommendation (Pan et al., 2005) with dynamic query generation. There are two independent recommenders in our system: a *retrieval-based recommender* and a *generation-based recommender*. The retrieval-based recommender produces recommendations by selecting from pre-stored examples while the generation-based recommender dynamically constructs recommendations based on the current user input. After combining the results from both, the hybrid recommender is able to recommend queries beyond the scope of the pre-stored examples. In the following, to explain the hybrid approach, first we summarize the retrieval-based approach, and then we focus on the new generation-based method.

3.1 Retrieval-based Recommendation

The main knowledge in retrieval-based recommendation is a query corpus containing typical query examples a system can understand. For each example in the corpus, the system stores not only the query itself, but also its semantics produced by a semantic interpreter as well as contextual features derived from the conversation history. When a problematic user query is encountered, the system searches for the most relevant examples from the corpus. The relevance of an example to a user query is defined as the weighted combination of three similarity scores: the string similarity score, the semantic similarity score and the context similarity score. Based on the total relevance score for each example in the corpus, the system will recommend the top N queries to the user.

This approach works effectively when the system stores almost all the valid query combinations in the query corpus. However, in any non-trivial conversation system, the number of queries a system can understand is huge due to combinatorial explosion. For example, in a real estate application in which there are 40 attributes associated with a house, a system would need to store at least 2^{40} examples just to cover queries related to searching for a house using any combination of the attributes as search constraints. Thus, it is often impossible to collect and pre-store all the queries ahead of time. To solve this problem, we propose to dynamically compose new queries based on the current user request so that the system can recommend queries beyond the scope of pre-stored examples.

3.2 Generation-based Recommendation

Dynamic query generation offers a significant advantage over the retrieval-based approach because the system is able to generate a large number of queries based on a small set of query examples. In general, it is difficult to cover all the combinations of concepts and attributes in a corpus. It is not difficult however, to cover each concept or attribute in at least one of the examples. In the following example, assume

there are seven concepts {A, B, C, D, E, F, G} and four query examples {ABE, CD, EFG, DFG}. When receiving a new user inquiry "ABDF□" in which "□" is unknown to the system, a retrieval-based recommender may not be able to produce any good recommendations because none of the existing examples is close enough to the user query. Using generation-based recommendation, however, the system can produce a query more similar to the user's problematic query. For example, based on the partially understood results and the system's guess that the meaning of "□" is "G", the system may decide to generate a sentence conveying "ABDFG" which is closer to the user query than any of the existing examples.

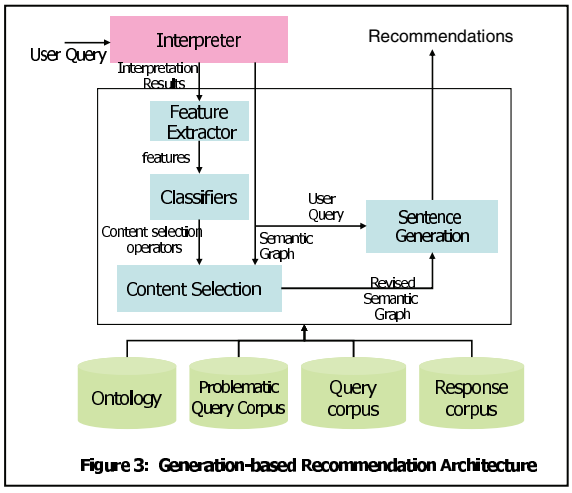

Figure 3: Generation-based Recommendation Architecture

To explain the generation-based recommendation method in detail, let's follow the flow in Figure 3. When a problematic user query, e.g. with unknown words, is received, the *interpreter* produces an error code plus partial interpretation results in the form of an incomplete semantic graph. From the user query and the partial interpretation results, the *feature extractor* automatically derives a set of syntactic and semantic features characterizing the unknown word in the user query. Based on the extracted features and annotated examples in the *problematic query corpus*, the *classifier* selects one or more content selection operators. Then the *content selector* revises the partial semantic graph using the selected operators to formulate the content of a recommendation. Finally, *the sentence generator* takes the revised semantic graphs and produces grammatical sentences as the recommendations. The following section explains this component in detail, starting with classification-based content selection.

3.2.1 Classification-based Content Selection

Content selection is critical in recommendation generation. It is an attempt for the system to restore the semantics of problematic user inputs based on the system's knowledge about the application domain, the current user input, and the conversation context. Depending on the type of interpretation errors, the system may add new content or remove existing content to produce a recommendation. Currently, we employ a classification-based approach to categorize the interpretation problems and to derive proper content selection operators. Since most problems that an interpreter can reliably identify involve unknown words, in this study, we focus on unknown word classification.

In the next section, we describe how to categorize each unknown word using a set of semantic and syntactic features. Later we explain how to build multiple classifiers to select appropriate content selection operators.

TABLE 1 FEATURES EXTRACTED FOR CONTENT SELECTION

No	Name	Definition	Example
1	EleSynRel	The relationship between the problem element and the element anchor	Modifier-Head
2	PrElePOS	The Part-of-Speech (POS) of the problem element	Adjective
3	PrEleRole	The role played by the problem element in its relationship with its anchor	Modifier
4	PrEle Classification	Further classification for the unknown word	NA
5	AncElePOS	The POS for the anchor element	NP
6	AncEleRole	The role played by the anchor element in its relationship with the problem element	Head
7	AncEleType	The semantic type associated with the anchor element	Known-Concept
8	PrSegSyn-Rel	The syntactic relation between the problem and the anchor segment	Modifier-Head
9	PrSegPOS	The POS for the problem segment	NP
10	PrSegRole	The role played by the problem segment in its relationship with its anchor	Modifier
11	AncSegPOS	The POS for the Anchor Segment	Noun
12	AncSegRole	The role played by the anchor segment in its relationship with the problem segment	Head
13	AncSegType	The semantic type associated with the anchor segment	Known-Concept

3.2.1.1 Classification Features

To categorize the problem associated with each unknown word and to derive proper content selection operators for recommendation, we extract a feature vector containing thirteen semantic and syntactic features for each unknown word. The rationale for selecting these features is to establish a connection between the unknown words and their context. If the system can understand some words that have direct relationships with the unknown word, the system may be able to infer the semantics of the unknown word. Most of these features are defined based on the following four concepts: *problem element*, *anchor element*, *problem segment* and *anchor segment*. A problem element is the basic token that contains the unknown word. An anchor element is a syntactic constituent that has a direct relationship with the problem element in a syntactic tree. For example, if the problem element is a modifier in a noun phrase, the anchor element will be the head of that noun phrase; or if the problem element is a syntactic object, the anchor element will be the verb of that object. Similarly, a problem segment is the most specific syntactic constituent that contains the problem element. The anchor segment is the closest syntactic constituent of the problem segment. We generally ignore function words when defining these concepts.

In the following example, "show houses in *unified* school district", we assume "*unified*" is the unknown word. Based on our definition, the problem element is "*unified*", element anchor is "*school district*"; problem segment is "*in unified school district*"; and anchor segment is "*houses*". Table 1 summarizes the definitions of the classification features.

3.2.1.2 Content Selection Operators

In the classification training corpus, after a feature vector is extracted for each unknown word in a sentence, we also assign one or more content revision operators for each vector, indicating proper ways to revise the partial understanding results to formulate the semantics of a recommendation. To define the set of content selection operators, we analyzed a set of problematic user queries. Table 2 listed four commonly used content revision operators in structured information seeking applications like real estate database search.

TABLE 2: CONTENT SELECTION OPERATORS

Name	Description	Example
Op-Constraint-Ontology	Replace an unknown word with constraints that are compatible with the word and the ontology	Replace "unified" in "*unified* school district" with all the school district constraints, e.g. Pleasantville school district.
Op-Constraint-Attribute	Replace an unknown word with constraints that are compatible with the current attribute	Replace "fair" in "Houses with *fair* tax" with a constraint on the house attribute: Annual *Tax,* e.g. tax less than $100000
Op-Attribute-Ontology	Replace an unknown word with attributes that are compatible with the ontology	Replace "dimension" in "the *dimension* of the house" with an attribute of house based on the ontology, e.g. square footage.
Op-Operator-Ontology	Replace unknown word with a known operator	Replace "preserve" in "preserve the houses" with a known operator like "save"

3.2.1.3 Classification Results

Our training examples for classification are collected from a wizard-of-oz (WOZ) user study. In total, we have collected 36 conversation segments and ~500 user requests. Among these requests, our system detected 187 unknown words. For each unknown word, a feature vector (described in Table 1) was extracted automatically from the interpretation results. In addition, for each feature vector, we manually assigned a *yes* or *no* tag for every content selection operator defined in table 2. In total, we have trained four content selection classifiers. Currently, we use JRip (Witten and Frank, 2005), a Java implementation of the RIPPER classifier (Cohen, 1998) in our experiment.

TABLE 3: CLASSIFICATION RESULTS

Classifier	Accuracy	Majority classification
OpConstraintOntology	98.4%	81.7%
OpConstraintAttribute	93.6%	83.9%
OpAttributeOntology	91.4%	81.8%
OpOperatorOntology	94.6%	78.6%

Table 3 shows the performance of each classifier based on ten-fold cross validation. We compare it with the performance of majority-based classifiers in which the classifiers always predict "*no*". The results indicate that content selection can quite reliably help the system in recovering the semantics of unknown words.

3.2.1.4 Applying Content Selection Operators

If a content selection operator is chosen, it is used to revise the semantic graph of the original user query. Three knowledge resources are used in this process: the domain ontology, the query corpus and the response corpus. For example, if the operator *OpAttributeOntology* is selected to revise the unknown word in "Show the xxx of the houses in Pleasantville", the system will retrieve all the attributes associated with the anchor concept "*House*" from the ontology. For each retrieved attribute, the system generates one semantic graph, resulting in many possible recommendations. In addition to the ontology, both the query corpus and the response corpus are also used when applying the other content selection operators. In the next example, if OpConstraintAttribute is chosen to revise the user query "show houses with *fair* tax" in which "*fair*" is the unknown word, the system will search both the query corpus and the response corpus to find all distinct query constraints that use the house attribute "annual tax", such as "houses with annual tax below $10000". Similarly, for each distinct constraint retrieved, the system generates one semantic graph.

After applying content selection operators, the results contain a set of semantic graphs, each representing the content of a recommendation. In the following section, we explain how to generate a sentence that not only conveys the semantics in the semantic graph faithfully but also is syntactically and lexically consistent with the original user query.

3.2.2 Cascade model for Sentence Generation

Once the content of a recommendation is determined, it is sent to the sentence generator to be realized as grammatical sentences. Here we implemented an instance-based sentence generator that selects and dynamically composes new sentences from examples in an instance corpus.

One critical issue in recommendation generation is to adapt a sentence's surface form to be as similar to the expressions in the original user input as possible so that it is easier for a user to identify the changes between them. For example, when the original user query "Show xxx houses in Pleasantville with 2000 sq ft." can not be understood by the interpreter, it is more desirable if the system recommends "Show 4 bedroom houses in Pleasantville with 2000 sq ft" than "Show 2000 sq ft houses with 4 bedrooms in Pleasantville" even though both convey the same semantics.

In the following, we describe an instance-based sentence generation approach that matches and selects words and phrases from a cascade of instance corpora so that the system can reuse as many expressions similar to the user's as possible. We start with a brief introduction on instance-based sentence generation.

3.2.2.1 Instance-based sentence generation

In instance-based sentence generation, all the semantic and syntactic knowledge needed for sentence generation is encoded in an instance corpus. Each instance in the corpus is

represented as a pair of semantic graph and realization tree. The semantic graph represents the concepts and relations conveyed in a sentence and the realization tree represents how the concepts and relations are realized using words and phrases. During generation, the input semantic graph from the content planner is matched against all the semantic graphs in the corpus and the closest matching one is retrieved with its associated realization tree. Moreover, if the matching is not perfect, based on the difference, a set of adaptation operators are derived so that new sentences can be generated by deleting unneeded content from, or adding new content to the associated realization tree. More details on instance-based sentence generation can be found in (Pan and Shaw 2004, Pan and Shaw 2005). In the following, we focus on the new cascade model which is designed to maximize the syntactic and lexical consistency between the original user query and the generated recommendations.

3.2.2.2 Cascade model for instance selection

The essence of instance-based sentence generation is to reuse as many words, phrases or even the entire sentences in the instance corpus to convey desired semantics. As a result, the size and style of the instance corpus can significantly impact the generation quality. Sufficient coverage ensures that there always exist proper words or phrases in the instance corpus to convey any given semantics. Using instances similar in style maximizes the chance for reusing large chunks of the corpus material in a new sentence. The larger the reused chunks are, the fewer adaptations are required, and the better the generation quality is.

To balance the needs for good coverage and similar style, we use three types of instances available in our system: the current *user input*, the examples in a *query corpus* and the examples in a *response corpus*. The current *user input* only contains one pair of semantic graph and realization tree automatically derived from the partial interpretation results. Since the system may make mistakes, it can be incorrect. However, it is the only source containing the content and form of the current user query. The *query corpus* contains typical queries our system understands. Each request in the *query corpus* is manually annotated with a semantic graph and a realization tree. Style-wise, it is similar to the recommendations to be generated. However, since the size of the query corpus is small[†], to improve the coverage, we also use the *response corpus*. The response corpus is designed initially for response generation. It is clean but with a somewhat different style. For example, query corpus contains users' requests for data, while instances in the *response corpus* are descriptions of query results. Nonetheless, they still share a significant amount of vocabulary.

In the cascade model, to ensure that the output sentences convey the desired semantics, instances are only selected from the two clean sources: the query corpus first, followed by the response corpus. But, to generate recommendations as close to the original user query as possible, we adapt the generation results based on the features extracted from the user query. Overall, three input features are extracted: the realization order, the realization form and the presence of discourse cue phrases. Table 5 shows the effects of adaptation using these features. In the first example, based on the interpretation results of Q_1, the system knows that in R_1 the house *bedroom* constraint should come before the *bathroom* and the *city* constraints. It is also aware that the bedroom and bathroom constraints should be realized as pre-modifiers instead of post-modifiers. Without taking input order and form into consideration, however, the system might generate a recommendation like R_2 which is considered worse than R_1. In the next example, assume Q_2 and Q_3 are two consecutive user queries. Since the system interprets user query in context, the interpretation results of Q_3 is equivalent to Q_3'. Without discourse cue phrase adaptation, the system will recommend R_3 which is less coherent and context appropriate than R_4.

TABLE 5 GENERATION WITH OR WITH OUT INPUT ADAPTATION

Q1: Show 3 bedroom 2 bathroom houses in xxx cities. **R1**: Show 3 bedroom 2 bathroom houses in cities with less than 1000 people. **R2:** In cities with less than 1000 people, show 2 bathroom houses with 3 bedrooms.
Q2 Show 3 bedroom houses **Q3**:Just those in xxx cities Q3' Show 3 bedroom houses in xxx cities **R3**:Show 3 bedroom houses in cities with less than 1000 people **R4**: Just those in cities with less than 1000 people

3.3 Merging Recommendations

After the retrieval-based and the generation-based recommender produce two sets of recommendations independently, we merge the results. Currently given the maximum number of recommendations to display on the recommendation panel, we take the number of results proportionally from each recommender. For example, if the maximum number of recommendations allowed is five, the generation-based recommender produces eight results, and the retrieval-based recommender produces two results, the final five recommendations contain four recommendations from the generation-based system, one result from the retrieval-based system.

4 Evaluations

We perform an evaluation to verify the usefulness of the proposed approach. Through this evaluation, we want to gather two results. First, we want to see whether the hybrid approach can improve recommendation quality over a baseline system. Second, we want to verify that recommendations dynamically generated by our approach are valid queries that can be understood by our system. Otherwise, users may be frustrated due to subsequent rejections of the recommendations by the interpreter.

In the first evaluation, we use the retrieval-based recommender as the baseline. Based on our previous study (Pan et al., 2005), systems aided by retrieval-based recommendation were more effective than the same system without any recommendations. Overall, users achieved higher task suc-

[†] There are 330 examples in the query corpus.

cess rate and also spent less time and fewer turns to complete their tasks. In addition, in their survey, users also reported that the system understood them better and they also had better idea on what to ask. To verify that the new hybrid system can perform better than the retrieval-based system, we collected about 850 unique queries from previous user study logs (not including the WOZ queries). Among them, 133 sentences contained unknown words. From them we randomly selected 50 test queries. For each test query, each system (hybrid or retrieval) generated a maximum of five recommendations (Top5). After mixing results from both systems and after redundancy removal, we randomly ordered all the recommendations and presented them to two human judges who had no knowledge about this work. For each recommendation, we asked them to decide whether the recommendation is useful for a given user request. A recommendation is useful if the judge knows how to revise the unknown words after seeing the recommendation. If a recommendation was selected by a judge, the approach that produced the recommendation scored one point. The system with higher overall score is the one that produced more useful recommendations. In addition, we also let the judges choose the best recommendation among all the useful ones (Top1). Table 6 summarizes the results. Based on the results, among the five recommendations produced for each user query, the hybrid approach produces 1.76 useful recommendations on average versus 1.16 by the retrieval-based approach. The difference is statistically significant using pair-wised *t* test. The hybrid system also performed significantly better than the retrieval-based system based on the Top1 evaluation result (0.56 versus 0.29). The difference is also statistically significant.

TABLE 6: EVALUATION RESULTS

Approach	Mean-Top5	Significant-Top5	Mean-Top1	Significant-Top1
Hybrid	1.76		0.56	
Retrieval	1.16	<0.001	0.29	<0.001

To verify whether the recommendations dynamically generated can be understood by the interpreter, we run all the generated recommendations through the interpreter. Our results show that 100% of them can be interpreted successfully. Overall, our evaluation indicates that the hybrid recommender not only improves the query recommendation quality by generating recommendations beyond the scope of pre-stored examples but also maintains the same level of query interpretability as the retrieval-based approach.

5 Conclusions

In this work, we designed and implemented an approach to recommend context appropriate query alternatives when a user's query cannot be understood by the system. We developed a hybrid recommendation strategy that extends the retrieval-based query selection with query generation. It provides a solution to one major problem in retrieval-based recommendation: scalability. This makes query recommendation more feasible for practical conversation applications. Moreover, since our approach dynamically generates recommendations on the fly, this makes it possible for the system to adapt the surface form of the recommendations so that they are lexically and syntactically consistent with the original user query. Our evaluation results confirmed the effectiveness of this approach.

References

S. Pan, S. Shen, M. Zhou, and K. Houck. 2005. Two-way adaptation for robust input interpretation in practical multimodal conversation systems. In Proc. of IUI, 35-42.

H. Hastie, M. Johnston and P. Ehlen. 2002. Context-sensitive help for multimodal dialogue. In Proc. of the International Conf. on Multimodal Interfaces.

A. Stein. 1998. Active help and user guidance in a multimodal information system: A usability study, In Report ABIS-98, 87-98, U.J. Timm and M Rössel (eds.).

C. Kamm, D. Litman and M. Walker. 1998. From novice to expert: the effect of tutorials on user expertise with spoken dialogue Systems. In Proc. of ICSLP, 1211-1214.

M. Walker, J. Fromer, G. Di Fabbrizio, G. Mestel and D. Hindle. 1998. What can I say? Evaluating a spoken language interface to Email. In Proc. of CHI, 582-589.

R. Barzilary and M. Lapata. 2005. Collective content selection for concept-to-text generation. In Proc. of HLT/EMNLP.

P. Duboue and K. McKeown. 2003. Statistical Acquisition of Content Selection Rules for Natural language generation. In Proc. of EMNLP

K. McKeown, S. Pan, J. Shaw, D. Jordan and B. Allen. 1997. Language generation for multimedia healthcare briefings. In Proc. of ANLP, 277-282.

M. Walker, O. Rambow and M. Rogati. 2002. Training a sentence planner for spoken dialogue using boosting, Computer Speech and Language, 16 (3):409-433.

I. Langkilde. 2000. Forest-based statistical sentence generation. In Proc. of the NAACL.

J. Shaw. 1998. Clause aggregation using linguistic knowledge. In Proc. of the INLG, 138-147.

S. Pan and J. Shaw.2004. SEGUE: A hybrid case-based surface natural language generator. In Proc. of INLG, 130-140.

S. Pan and J. Shaw 2005. Instance-based Sentence Boundary Determination by Optimization for Natural Language Generation. In Proc. Of ACL. 565-572

W. Cohen. 1995. Fast effective rule induction. In Proc. of the International Conf. on Machine Learning 115-123.

I. Witten and E. Frank. 2005. Data mining: practical machine learning tools and techniques. Morgan Kaufmann.

Google. 2006. Google spell check. http://www.google.com/features.html#spell

M. Zhou, K. Houck, S. Pan, J. Shaw, V. Aggarwal, Z. Wen: Enabling context-sensitive information seeking. In Proc. of Intelligent User Interfaces 2006. 116-123