

Learning Database Content for Spoken Dialogue System Design

Joseph Polifroni and Marilyn Walker

The Department of Computer Science
University of Sheffield
Regent Court
Sheffield, S1 4DP
UNITED KINGDOM
j.polifroni@dcs.shef.ac.uk/m.a.walker@sheffield.ac.uk

Abstract

Spoken dialogue systems are common interfaces to backend data in information retrieval domains. As more data is made available on the Web and IE technology matures, dialogue systems, whether they be speech- or text-based, will be more in demand to provide user-friendly access to this data. However, dialogue systems must become both easier to configure, as well as more informative than the traditional form-based systems that are currently available. We present techniques in this paper to address the issue of automating both content selection for use in summary responses and in system initiative queries.

1. Introduction

One of the most common applications for spoken dialogue systems is as an interface to structured databases, where database entities are represented in terms of a set of attributes and their values. For example, in the restaurant domain, attributes for the restaurant entity include *cuisine*, *neighborhood*, *food quality*, *price* and *neighborhood* (Polifroni et al., 2003; Walker et al., 2004). To provide access to such databases, many spoken dialogue systems are configured to have an initial information-gathering phase, where the values for certain attributes are elicited from the user. The attributes (also referred to as "constraints"), as well as a default order of elicitation, are specified by a system developer familiar with the domain. The goal of the information gathering phase is to reduce the number of database tuples to a subset of the data that can be easily described. Any information-providing utterances by the system are thus delayed until the system has elicited enough constraints to query the database. Table 1 shows an example of such a system, modelled after the strategy described in (Levin et al., 2000). Note that user utterances User1, User2, and User3 successively narrow the restaurant selections, until there are only two matching tuples in the database.

Furthermore, as shown by utterance System4, the information-providing utterances in such systems usually take the form of descriptions of individual entities within the database. The attributes to be mentioned in these descriptions are also specified by a system developer familiar with the domain.

Dialogue systems designed in this way have several serious limitations. First, users may wish to *browse* the data, either because they are unfamiliar with the domain or because they do not have strong preferences. Second, this type of dialogue interaction often leads to the generation of overspecific constraints with unsatisfiable queries. Third, a default order of constraint elicitation may encourage the user to provide information that will not narrow down their choices in the most efficient way.

Our hypothesis is that these limitations can be addressed in

User1:	I'd like to know about restaurants in London.
System1:	Are you interested in a particular neighborhood?
User2:	I'd like an expensive restaurant.
System2:	Are you interested in a particular neighborhood?
User3:	I'd like an Italian restaurant.
System3:	Are you interested in a particular neighborhood?
User4:	Do you know of any near Leicester Square?
System4:	I know of the following two Italian restaurants near Leicester Square. Quo Vadi, on Dean Street, serves contemporary Italian cuisine. Signor Zilli, on Dean Street, serves traditional Italian cuisine.

Table 1: An example dialogue showing a traditional spoken dialogue management strategy, where constraints are elicited in a particular order (which the user is free to ignore) and no information is given to the user until a small number of database tuples is identified.

a domain independent way by applying statistical learning techniques to the automation of constraint order elicitation and content selection. We have applied these techniques to dialogue management and the resulting data-driven system responses are illustrated in Table 2. Section 2 describes in a general way the basic idea of how to automatically generate such responses using entropy and tree-induction. Section 3 describes the generation algorithms in more detail in the context of *Bruiser* (Browsing Restaurants Using Induction and Selection with Entropy), a spoken dialogue system targeting the London restaurant domain. Section 4 describes an exploratory experiment to evaluate these techniques. Section 5 discusses related work and section 6 summarizes and discusses future work.

User1:	What do you know about restaurants in London?
System1:	There are 592 restaurants in London in total. Most are expensive There are medium-priced restaurants near South Kensington tube station and in Chelsea. Is there a tube station or street you're interested in?
User2:	How about Soho?
System2:	Soho has many medium-priced restaurants with good food quality. I know of some expensive restaurants with very good food quality near Leicester Square tube station. Do you have a preference for cuisine?

Table 2: Example dialogue responses automatically derived using tree-induction and entropy.

2. Generating responses in dialogue systems using Entropy and Tree Induction

We want to be able to generate two different types of responses, intensional summaries and data-specific queries in a data-specific and domain-independent way. Intensional summaries allow the system to speak about the domain data at an abstract level of generalization by using intensional knowledge. These summaries require selection of appropriate pieces of information from the backend database to present to the user.

An example of intensional summary responses can be seen in Table 2. The two responses contain no extensional information at all; no specific instances of database tuples are contained within it. However, the answers do summarize the existing tuples into a statement containing information about their neighborhoods and price ranges.

The second type of response we are interested in generating is a data-specific system initiative (question). For example, if a user is browsing restaurants in Boston and has told the system s/he is interested in the North End neighborhood of the city, a follow-on query about cuisine would be almost redundant—89% of the restaurants in the North End are Italian. A better strategy might be to alert the user to this fact in a summary response and ask for another attribute in the system initiative. Although the user is free to ignore the system initiative and ask about any other attribute that the system knows about, the system initiative provides a useful hint, especially for new users, about what constraints the system knows about and how the user might proceed.

To generate these types of responses, the dialogue manager must query the database at *every* turn in the dialogue where the user has specified or refined a set of preferences. The database is filtered based on the user preferences and the resulting tuples are used to determine the content of the summary and system initiative responses. Both response types make use of a measure of entropy, or information gain.

In a multivariate space, entropy calculates the amount of information that is gained by choosing a particular path at any given point in the exploration of that space (Witten and Frank, 2000). Entropy will tend to score highly those attributes that have a large number of distinct val-

ues. For example, in a restaurant domain, the attribute *street* will typically have a larger entropy value than the attribute for *neighborhood*, given that most cities have a far greater number of unique streets than unique neighborhoods.

To automatically learn potential summary information about the backend, and enable more descriptive answering behavior, we use tree induction to discover rules about the data in focus at each turn in the dialogue. Decision tree induction has long been used in data-mining and has been shown to be an efficient way to compute association rules among attributes (Kamber et al., 1997; Liu et al., 2000). Our main purpose in applying tree induction to spoken dialogue systems was as a way of learning generalizations over our backend data tuples that could serve as useful summaries. Because the purpose of any tree-induction algorithm is to discover those association rules with the highest coverage, the resulting rules are used to form the basis for system responses.

The specific instantiation of the tree-induction algorithm used here is C4.5 (Quinlan, 1993). C4.5 is robust in cases where there is missing data among the attributes to consider, and it is able to deal with both scalar and symbolic values for attributes. Both of these were important to insure that the technique will be applicable to multiple domains. Furthermore, c4.5 performs quite efficiently, meaning that association rules for most simulated dialogue data can be computed while preserving the requirement for real-time dialogue performance.

3. Implementation in Bruiser

We have designed and implemented a dialogue manager that makes use of entropy and tree induction to produce responses in the Bruiser system, targeting the London restaurant domain. Control is table-driven and proceeds in sequence through a set of operations (Seneff et al., 1999) governing a four-step process:

- Filtering the data to match user constraints
- Computing entropy on the resulting data
- Generating association rules using tree induction
- Selecting from the association rules an appropriate set for presentation to the user

3.1. Filtering data

The first step involves filtering database tuples to match the user's constraints as specified up to that point in the dialogue. In our system, the dialogue manager interacts with the backend database to filter data. All subsequent decisions on what to speak are based on the subset of the data.

3.2. Computing entropy

If the resulting subset of the data is too large to speak to the user, the dialogue manager invokes the second step, calling a separate server whose job is to compute entropy on the values in the filtered data. The entropy values are stored by the dialogue manager. Entropy scores are used in two ways by the dialogue manager. The first is in constructing a system initiative.

As mentioned above, we wish to produce a system initiative at every point in the dialogue. A menu-driven system initiative strategy works through a set of constraints in a pre-determined order in every dialogue. To maintain an automated and data-driven dialogue management, the dialogue manager chooses the value with the highest entropy and enters it into the response frame being constructed. The natural language generation component of the system (Baptist and Seneff, 2000) will later construct a query to the user to determine if s/he has a preferred value for that attribute. Entropy values for a set of attributes representing the entire set of London restaurants are shown in Table3. As mentioned above, attributes with many distinct values score higher for entropy.

By using the attribute with the highest entropy value as the system initiative (e.g., *street* in Table3 for all restaurants), we reasoned that we could query the user for the value that would reduce the search space most effectively. Because an individual attribute may continue to appear at the top of the entropy measure until a user has specified a value for it, our system maintains a record of attributes it has asked about. If an attribute has been asked about already in the dialogue, the system will choose, in the following turn, the attribute with the next entropy value to use in the system initiative.

3.3. Generating association rules using tree induction

In the third step of the content selection process, tree-induction is performed by another server using the C4.5 algorithm. Trees are generated iteratively, beginning with a randomly selected subset of the data. Multiple trees are built and later pruned to simplify the trees. Entropy values, calculated earlier, are sent to the C4.5 server and used in the construction of the trees.

Trees are built to predict a dependent variable (i.e., the leaf node) and we use the entropy values to select the dependent variable for each tree. We experimented with various ways of using entropy and finally decided on using the attribute with the lowest entropy value (e.g., *price_range* in Table 3 for the set of all restaurants). When we used values with high entropy, we found that the rules were far too specific. For example, the attribute *street* often has the highest entropy value, as evidenced from the values for entropy computed for the first query in Table3. A tree designed to predict a set of restaurants based on *street*, however, finds few clusters of restaurants. (Although there are many streets in London with multiple restaurants, association rules that predict *street* in combination with other attributes find few restaurants.) In a browsing domain, attributes with *high* entropy actually make for the best generalizations. When there are just four possible values for a particular attribute, as is the case for price range (quantized in our implementation into bins representing very expensive, expensive, medium priced, and cheap), association rules find larger clusters of restaurants to speak about.

3.4. Selecting from the association rules for presentation to the user

Each set of association rules in the tree is given a score representing the number of database tuples it correctly identifies minus the number of tuples that it incorrectly identifies,

through multiple iterations. The resulting rules are ordered by score and returned to the dialogue manager for the fourth and final step of content selection.

The dialogue manager examines the association rules and determines the best set to send on to the natural language generation module. Currently, we are only looking at the best-scoring rules, i.e., the rules that account for the most data. The top-scoring rule is always selected and subsequent rules are examined to find a set that both accounts for a large number of tuples and includes a range of predicate types. The system is configured to choose at most three rules, for reasons of conciseness. If adjacent rules account for the same predicate types, preference is given to lower scoring rules that describe different predicates. A thresholding parameter gives precedence to rules whose scores are close in value and penalizes rules whose scores are very different from the immediately preceding one. Once rules are selected by the dialogue manager, predicates with identical values are aggregated to make the resulting response string less verbose. In order to avoid annoying the user by continually asking for the same attribute in the system initiative, the dialogue manager, as a final step, examines its history of system initiative attributes and may decide to choose one with a lower entropy value if the user seems uninterested in the one that has been asked about.

Table 2 shows an example of system behavior using the algorithms described above. With no constraints on the data (all restaurants are in London), the database attribute with the lowest entropy value was *price_range*, which was used as the dependent variable in constructing the tree used in content selection. The first sentence in the response (System1) is a generated form of the top-scoring rule, which included the single attribute *price_range*. The second sentence is a representation of the next two rules, chosen because they have a similar score and represent different sets of predicates (i.e., *tube_station* and *neighborhood*). Since both rules have the same value for “price,” it is spoken just once in the resulting string. The rules used to construct this response are shown in Table4.

The attribute from the dataset with the highest scoring entropy value was *street*. In cases where entropy values at the upper end are close, a heuristic is employed that takes into account the individual scores of the entropy values as well as their spread to enable a system initiative to be constructed using the top two attributes. The response marked System1 in Table 2 shows an example of this, where the system initiative offers the user a choice between *tube_station* or *street*. In the example in Table 2, the simulated user ignores the system initiative and introduces a constraint on *neighborhood* in the second query (User2).

Filtering the data on the *neighborhood* constraint, we calculate entropy on the new data set, and present those data to the tree-induction algorithm. The resulting set of rules in Table5 was used to construct the response marked System2. In this case, only two rules were selected for natural language generation because the number of tuples accounted for by the third rule fell below the selection threshold. *Food_quality*, the attribute with the lowest entropy value among the restaurants in Soho, was used as the leaf node in tree construction.

Attribute	Entropy value
street	1.5114
tube_station	1.23
food_range	0.6947
cuisine	0.4719
neighborhood	0.30702
price_range	0.2252
Attribute	Entropy value
cuisine	1.372
street	1.014
tube_station	0.785
foodrange	0.49

Table 3: Example entropy values computed for the set of all London restaurants in our corpus (top table) and for restaurants in Soho (bottom table).

--> price = expensive (330)
:tube = "South Kensington"
--> price = "medium" (20)
neighborhood = "Chelsea"
--> price = "medium" (20)
:tube = "Piccadilly Circus"
--> price = "expensive" (18)
:food = "good" && :street = "Gerrard"
--> price = "expensive" (15)

Table 4: Example rules derived from the C4.5 tree-induction algorithm applied to a database of London restaurants. The number in parentheses indicates the number of restaurants accounted for by the rule.

Table 6 shows another example of a dialogue with responses derived using entropy and tree induction. In this dialogue, the user has begun by specifying restaurants in South London. Association rules cluster those restaurants by price_range in the first utterance in System1 and by neighborhood and food_quality in the second utterance in System1. A follow-up query with a constraint on food_quality results in a response grouping tube_station and price_range in the response (System2 in Table 6). The system initiative queries are different in this dialogue as compared to that in Table 2. Although street appears in both dialogues in the first query, food_quality is also used in the first system initiative in the dialogue in Table 6, compared with tube_station, in the first system query in Table 2.

:food = good --> :price = medium (12)
:tube = Leicester Square && :food = medium
--> :price = expensive (8)
:cuisine = Indian && :food = good
--> :price = medium (4)
:tube = Oxford Circus && :food = very good
--> :price = expensive (4)

Table 5: Example rules derived from the C4.5 tree-induction algorithm applied to a database of London restaurants.

4. Evaluation

We are currently in the process of conducting experiments to determine user preferences among three different systems within the London restaurant domain. The first is the *Bruiser* system described in this paper, using machine learning techniques to derive content for summarization and system initiative queries. The second system is a simple baseline system, as illustrated in Table 1, which we refer to as the *DirectedDialogue* system. This system elicits constraints in a predetermined order by asking for a particular constraint that is absent from the underlying form-based representation. The third system, which we call *DataDriven1*, is briefly described below.

DataDriven1 is based on previous work which sought to make dialogue design more data-driven and to generalize more dialogue parameters with less input from system developers (Polifroni et al., 2003). *DataDriven1* is similar to *Bruiser* in that content selection for summarization is determined by an analysis of the backend data, filtered at each turn to match the user constraints. Each attribute is examined and the set whose top N values accounted for $M\%$ of the data is chosen to speak, where N and M are thresholds set by the system developer *a priori* for each attribute in the database. Although this provides a mechanism for describing the backend data, content selection is still hand-crafted to the extent of defining thresholds and, perhaps more critically, associations between attributes are left unexplored. Table 7 shows an example of output derived using the *DataDriven1* system and *Bruiser*.

There are two important distinctions between *Bruiser* and *DataDriven1*. The first is that *DataDriven1* requires input from the system developer in the form of thresholds for speaking. For certain subsets of the data, these thresholds can result in no attributes being selected for speaking. *Bruiser* does not require such thresholds and, furthermore, always finds something to speak about in the backend data. The second distinction is that, in the *DataDriven1* approach, each attribute is evaluated individually and the resulting responses speak about the attributes in individual utterances. Although both systems may choose to speak about the same set of attributes, the *Bruiser* system, using tree induction to discover association rules, allows us to speak about relationships among attributes, which gives the user a better understanding of the domain.

An examination of the two dialogues shown in Table 7 illustrates these differences. Both dialogues are driven by the same simulated user utterances. The dialogue at the top of Table 7, the one using the *DataDriven1* approach, has terser responses that contain less information about the backend data. The dialogue at the bottom of Table 7, using the *Bruiser* system, shows the result of learning association rules from the data (e.g., the relationship between price_range *expensive* and tube_station in response "Bruiser-1", and the relationship among price_range, neighborhood, and area of London in response "Bruiser-2"). One other distinction is that the *DataDriven1* approach did not produce a system initiative query for each response. As with the *DirectedDialogue* system both of these systems simply enumerate the set of tuples when that set contains five or fewer members.

User1:	I'd like to know about restaurants in South London?
System1:	There are 178 restaurants in South London in total. There are 74 expensive ones. There are 8 in Chelsea with good food quality. Do you have a preference for street or food quality?
User2:	I'd like good food quality.
System2:	There are 52 restaurants with good food quality in south London. There are 9 medium priced restaurant near the South Kensington tube station. There are 6 expensive restaurants near the Sloane Square tube station. Do you have a preference for cuisine?

Table 6: Another dialogue pair showing responses automatically derived using tree-induction and entropy.

All three systems use the same architecture, giving us a unique opportunity to derive dialogues using the same input queries and backend data. We are performing these experiments in an “overhearer” mode, where judges are presented with three separate responses for each user query and asked to rate each response.

An exploratory study showed that users prefer either the *DirectedDialogue* baseline system or the *Bruiser* system. We were surprised at the number of people who preferred the directed dialogue approach, but feel that this is in keeping with the assumption that successful dialogue strategies are dependent on user preferences and knowledge (Kass and Finin, 1988). In cases where a user is unfamiliar with the knowledge contained in the backend database, responses that provide more information might be preferred. If users are familiar with the data, they may feel that a directed dialogue strategy will more efficiently allow them to reach their goals. Individual differences related both the user’s knowledge and their “degree of hastiness” (Komatani et al., 2003) could also play a role in the perceived efficacy of the strategy types. We plan to run more experiments to determine the effect of these dialogue strategies on user satisfaction.

5. Related work

The summarization strategy here differs considerably from the view of a summary as a condensed version of an original document or set of documents. In this respect, our strategy is closer to indicative summarization. Indicative summarization (Kan et al., 2001b) has been used to provide insight in the contents of documents and help guide the user to a specific document that can then either be summarized using other techniques, or retrieved as a whole. Features of individual documents are examined to determine useful summaries, in much the same way that features of individual database tuples can be examined to determine useful aspects to summarize. Tree structures have been used to characterize topic information, although these tree structures have been determined using section headers, for example, in structured documents (Kan et al., 2001a). One important difference is our use of automatically computed *association* rules among tuples in the database to give users insight into generalizations that may be inherent in the data but not obvious to even someone experienced with the domain.

Another motivation in adopting this approach to content selection comes from the field of cooperative answering.

The ability to process existing data to enable generalization and summarization has been proposed as a mark of intelligent answering (Han et al., June 1996), as well as the ability to browse the data (Motro, 1996). The user should be able to explore what is contained within a backend without necessarily having a clear goal in mind. The techniques described in this paper automate these capabilities for dialogue systems.

The field of data visualization contains unexpected similarities with dialogue system design (for database access), in that both concern themselves with making sense of large corpora of data with the purpose of providing a sensible interface to users. Early work in this field (A. Buja and Stuetzle, 1991; Buja et al., 1996) concerned itself with selecting subsets of the data to focus users on particular aspects of a problem. Later work treats the problem of data visualization as one of data *summarization* (Lesh and Mitzenmacher, 2004), as we do here. The emphasis in much of the data visualization literature is in the interface to the user, including an iterative interaction via a GUI that seems very like a dialogue. Our work complements this approach by adding a generation component for a natural language rendering of the summarizations, as well as a speech and natural language input capability.

6. Summary and future directions

We have reported here on our initial attempts at using machine learning techniques for automatically determining both summary attributes and system initiative for dialogue system responses. We have implemented these techniques in the context of a restaurant domain for London and are currently running an evaluation experiment. In future work, we will expand the dialogue manager to incorporate these algorithms as part of a strategy for generating fallback responses in cases where the database returns no tuples matching the user’s request.

Since our original motivation was to provide useful and helpful information to the user, user modelling will fit in well with our goals. We intend to examine the application of user modelling to help guide content selection from both the entropy values and the association rules we derive from our dialogue manager.

These techniques aim to produce ways of using an attribute/value data representation to construct responses in dialogue systems. We are, therefore, exploring ways that the same algorithms that work for the restaurant domain could be applied to different domains. In particular, we

User1:	I'm interested in expensive restaurants in London.
DataDriven1-1:	I know of 320 expensive restaurants in London. There are 45 choices for cuisine.
User2:	I'd like an Italian restaurant.
DataDriven1-2:	There are 73 expensive Italian restaurants in total. They are predominantly traditional and contemporary.
User3:	Do you know of any near the Leicester Square tube station?
DataDriven1-3:	I know the following two expensive Italian restaurants near Leicester Square tube station
User1:	I'm interested in expensive restaurants in London.
Data-Driven2-1:	I know of 320 expensive restaurants in London. There are 13 near the Knightsbridge tube station and 8 near the Farrington tube station. There are also some in Saint James's. Do you have a preference for street or tube station?
User2:	I'd like an Italian restaurant.
DataDriven2-2:	There are 73 expensive Italian restaurants in total. There are 8 in central London in Knightsbridge and 7 in South London in Chelsea. Do you have a preference for a neighborhood?
User3:	Do you know of any near the Leicester Square tube station?
DataDriven2-3:	I know the following two expensive Italian restaurants near Leicester Square tube station

Table 7: Example dialogue responses illustrating the differences between the *DataDriven1* approach to content selection and response generation (at top) and the *Bruiser* system (at bottom).

are interested in domains both different in scope from traditional data-retrieval domains and with a compelling reason for a dialogue interface. We are currently conducting a preliminary study using the techniques described above to simulate a dialogue interface to a large corpus of annotated news articles.

7. References

- J. Michalak A. Buja, J. A. McDonald and W. Stuetzle. 1991. Interactive data visualization using focusing and linking. In *Proc. of IEEE Visualization '91*, pages 156–163, San Diego, California.
- L. Baptist and S. Seneff. 2000. Genesis-ii: A versatile system for language generation in conversational system applications. In *Proc., ICSLP 2000*, Beijing.
- A. Buja, D. Cook, and D. Swayne. 1996. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5:78–99.
- J. Han, Y. Huang, N. Cercone, and Y. Fu. June, 1996. Intelligent query answering by knowledge discovery techniques. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):373–390.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J Han. 1997. Generalization and decision tree induction: efficient classification in data mining. In *Proc. 7th International Workshop on Research Issues in Data Engineering (RIDE '97)*, pages 111–121.
- M.-Y. Kan, K.R. McKeown, and J. Klavans. 2001a. Applying natural language generation to indicative summarization. In *Proceedings of the Eighth European Workshop on Natural Language Generation*.
- Min-Yen Kan, Kathleen R. McKeown, and Judith L. Klavans. 2001b. Domain-specific informative and indicative summarization for information retrieval. In *Proceedings of the Document Understanding Conference*, New Orleans, U.S.A.
- R. Kass and T. Finin. 1988. Modelling the user in natural language systems. *Computational Linguistics*, 14:5–22.
- K. Komatani, S. Ueno, T. Kawahara, and H. Okuno. 2003. Flexible guidance generation using user model in spoken dialogue systems. In *Proc. ACL*, pages 256–263, Sapporo, Japan.
- Neal Lesh and Michael Mitzenmacher. 2004. Interactive data summarization: an example application. In *Proc., Working Conference on Advanced Visual Interfaces*, pages 183–187, Gallipoli, Italy.
- E. Levin, S. Narayanan, R. Pieraccini, K. Biatov, E. Boccieri, G. Di Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker. 2000. The attarpa communicator mixed-initiative spoken dialog system. In *Proc. ICSLP*, Beijing.
- B. Liu, Y. Xia, and P. Yu. 2000. *Clustering through decision tree construction*. IBM Research Technical Report RC21695.
- Amihai Motro. 1996. Cooperative database systems. *International Journal of Intelligent Systems*, II:717–731.
- J. Polifroni, G. Chung, and S. Seneff. 2003. Towards the automatic generation of mixed-initiative dialogue systems from web content. In *Proceedings of Eurospeech*, pages 193–196.
- J.R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.
- S. Seneff, R. Lau, and J. Polifroni. 1999. Organization, communication, and control in the galaxy-ii conversational system. In *Proc. Eurospeech '99*, pages 1271–1474, Budapest.
- M. Walker, S.J. Whitaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. 2004. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28:811–840.
- I. Witten and E. Frank. 2000. *Data Mining*. Morgan Kaufmann.