NATURAL LANGUAGE CALL ROUTING: A Robust, Self-Organizing Approach

Bob Carpenter *Jennifer Chu-Carroll*

Lucent Technologies Bell Laboratories
600 Mountain Avenue, Murray Hill, NJ 07974, U.S.A.

ABSTRACT

We have developed a domain independent, automatically trained, call router which directs customer calls based on their response to an open-ended "How may I direct your call?" query. Routing behavior is trained from a corpus of transcribed and hand-routed calls and then carried out using vector-based information retrieval techniques. Terms consist of sequences of morphologically reduced content words. Documents representing routing destinations consist of weighted term frequencies derived from calls to that destination in the training corpus.

In this paper, we evaluate our approach in the context of a large financial services call center with thousands of possible customer activities and dozens of routing destinations. We evaluate the system's performance on ambiguous and unambiguous calls when given either accurate transcriptions or fairly noisy real-time speech recognizer output. We conclude that in a highly complex call center, our system performs at roughly the same level of accuracy as human operators.

1. INTRODUCTION

The *call routing* task involves directing a user's call to the appropriate destination within a call center or providing some simple information, such as loan rates. In current systems, the user's goals are typically gleaned via a touch-tone system employing a rigid hierarchical menu. The primary disadvantages of navigating menus for users are the time it takes to listen to all the options and the difficulty of matching their goals to the options; these problems are compounded by the necessity of descending a nested hierarchy of choices to zero in on a particular activity. Even simple requests such as *"I'd like my savings account balance"* may require users to navigate as many as four or five nested menus with four or five options each. We have developed an alternative to touch-tone menus that allows users to interact with a call router in natural spoken English dialogues just as they would with a human operator.

Human operators respond to a caller request either by routing the call to an appropriate destination, or by querying the caller for further information to determine where to route the call. Our automatic call router has these two options as well as a third option of sending the call to a human operator. The rest of this paper provides both a description and an evaluation of an automatic call router driven by vector-based information retrieval techniques. After introducing our fundamental routing technique, we focus on the impact of speech recognition on performance. In other papers, we provide details of the speech recognizer (Reichl et al. 1998) and the disambiguation module (Chu-Carroll and Carpenter 1998). The main advantages of our system are that 1) it is domain independent, 2) it is trained fully automatically to both route and disambiguate requests, and 3) in contrast to touch-tone solutions, it performs at roughly the level of accuracy and efficiency of human operators.

	Name	Activity	Indirect
# of calls	949	3271	277
% of all calls	21.1%	72.7%	6.2%

Table 1: Semantic Types of Caller Requests

2. RELATED WORK

Call routing is similar to topic identification (see McDonough et al. 1994) and document routing (see Schütze et al. 1995) in identifying which one of n topics (destinations) most closely matches a caller's request. Call routing is distinguished from these activities by requiring a single destination, but allowing a request to be refined in an interactive dialogue.

The only work on natural language call routing to date that we are aware of is that by Gorin et al. (1997). They select salient phrase fragments from caller requests, such as *"made a long distance"* and *"the area code for"*. These fragments are used to determine the most likely destination(s) for the request either by computing the *a posteriori* probability for each call type or by passing the fragments through a neural network classifier.

3. CORPUS ANALYSIS

We analyzed a set of 4497 transcribed telephone calls involving customers interacting with human operators at a large call center that provides financial services in hundreds of categories in the general areas of banking, credit cards, loans, insurance and investments; we concentrated on the 23 destinations for which we had at least 10 calls in the corpus.

The operator provides an open-ended prompt of *"How may I direct your call?"* We classified user responses into three categories. First, callers may explicitly provide a **destination name**, either by itself or embedded in a complete sentence, such as *"may I have consumer lending?"*. Second, callers may describe the **activity** they would like to perform. Such requests may be unambiguous, such as *"I'd like my checking account balance"*, or ambiguous, such as *"car loans please"*, which in our call center can be resolved to either *consumer lending*, which handles new car loans, or to *loan services*, which handles existing car loans. Third, a caller can provide an **indirect request**, in which they describe their goal in a roundabout way, often including irrelevant information. Table 1 shows the distribution of caller request in our corpus. For the vast majority of calls, the request was based on destination name or activity. Our strategy was to detect and reject indirect queries and either re-prompt or route them to a human operator for handling.

We also analyzed the operator's responses to caller requests to determine the dialogue actions needed for response generation in our automatic call router. We found that in the call routing task, the call operator either *notifies* the customer of the routing destination or asks a disambiguating *query*. Table 2 shows the frequency that each dialogue action should be employed based strictly on the presence of ambiguity in the caller requests in our corpus. We

	Notification	Query	
		NP	Others
# of calls	3608	657	232
% of all calls	80.2%	14.6%	5.2%

Table 2: Call Operator Dialogue Actions

further analyzed those calls considered ambiguous within our call center and noted that 75% of such ambiguous requests involve an underspecified noun phrase (NP), such as requesting *car loans* without specifying whether it is an *existing* or *new* car loan. The remaining 25% of the ambiguous requests involve underspecified verb phrases, such as asking to *transfer funds* without specifying the types of accounts to and from which the transfer will occur, or missing verb phrases, such as asking for *direct deposit* without specifying whether the caller wants to *set up* or *change an existing* direct deposit.

4. TRAINING

Our training corpus consists of 3753 calls each of which is hand-routed to one of 23 destinations.[1] Our first step is to create one (virtual) document per destination, which contains the text of the callers' contributions to all calls routed to that destination.

We filter each (virtual) document through the morphological processor of the Bell Labs' Text-to-Speech synthesizer (see Sproat, ed. 1998) to extract the root form of each word in the corpus. Next, the root forms of caller utterances are filtered through two lists, the *ignore list* and the *stop list*, in order to build a better n-gram model. The ignore list consists of noise words, such as *uh* and *um*, which sometimes get in the way of proper n-gram extraction, as in *"I'd like to speak to someone about a car uh loan"*. With noise word filtering, we can properly extract the bigram *"car,loan"*. The stop list enumerates words that do not discriminate between destinations, such as *the*, *be*, and *afternoon*. We modified the standard stop list distributed with the SMART information retrieval system to include domain specific terms and proper names that occurred in the training corpus (see Salton 1971). Note that when a stop word is filtered out of the caller utterance, a placeholder is inserted to prevent the words preceding and following the stop word to form n-grams. For instance, after filtering the stop words out of *"I want to check on an account"*, the utterance becomes *"<sw> <sw> <sw> check <sw> <sw> account"*. Without the placeholders, we would extract the bigram *"check,account"*, just as if the caller had used the term *checking account*. We extract the n-gram terms that occur more frequently than a pre-determined threshold and do not contain any stop words. Our current system uses unigrams that occurred at least twice and bigrams and trigrams that occurred at least three times in the corpus. No 4-grams occurred three times. Employing this strategy, we found 420 unigram terms, 275 bigram terms, and 62 trigram terms.

Once the set of relevant terms is determined, we construct an $m \times n$ term-document frequency matrix A whose rows represent the m terms, whose columns represent the n destinations, and where an entry $A_{t,d}$ is the frequency with which term t occurs in calls to destination d.

It is often advantageous to weight the raw counts to fine tune the contribution of each term to routing. We begin by normalizing the row vectors representing terms by making them each of unit length. Thus we divide each row A_t in the original matrix by its length, $(\sum_{1 \le e \le n} A_{t,e}^2)^{1/2}$. Our second weighting is based on the notion that a term that only occurs in a few documents is more important in discriminating among documents than a term that occurs in nearly every document. We use the *inverse document frequency (IDF)* weighting scheme (see Sparck Jones 1972), under which a term is weighted inversely to the number of documents in which it occurs, by means of $IDF(t) = \log_2 n/d(t)$ where t is a term, n is the total number of documents in the corpus, and $d(t)$ is the number of documents containing the term t. Thus we obtain a weighted matrix B, whose elements are given by $B_{t,d} = A_{t,d} \times IDF(t)/(\sum_{1 \le e \le n} A_{t,e}^2)^{1/2}$.

To reduce the dimensionality of our vector representations for terms and documents and cast them into the same vector space, we applied the singular value decomposition to the $m \times n$ matrix B of weighted term-document frequencies (see Deerwester et al. 1990). Specifically, we take $B = USV^{\mathrm{T}}$, where U is an $m \times r$ matrix (where r is the rank of B), V is an $n \times r$ matrix, and S is an $r \times r$ diagonal matrix such that $s_{1,1} \ge s_{2,2} \ge \cdots \ge s_{r,r} > 0$. We think of each row in U as an r-dimensional vector that represents a term, whereas each row in V is an r-dimensional vector representing a document. With appropriate scaling of the axes by the singular values on the diagonal of S, we can compare documents to documents and terms to terms using their corresponding points in this new r-dimensional space (see Deerwester et al. 1990). For instance, to employ the dot product of two vectors as a measure of their similarity as is common in information retrieval (see Salton 1971), we have the matrix $B^{\mathrm{T}}B$ whose elements contain the dot product of document vectors. Because S is diagonal and U is orthonormal, $B^{\mathrm{T}}B = VS^2V^{\mathrm{T}} = VS(VS)^{\mathrm{T}}$. Thus, element i, j in $B^{\mathrm{T}}B$, representing the dot product between document vectors i and j, can be computed by taking the dot product between the i and j rows of the matrix VS. In other words, we can consider rows in the matrix VS as vectors representing documents for the purpose of document/document comparison. An element of the original matrix $B_{i,j}$, representing the degree of association between the ith term and the jth document, can be recovered by multiplying the ith term vector by the jth scaled document vector, namely $B_{i,j} = U_i((VS)_j)^{\mathrm{T}}$.

5. ROUTING

Our call router consists of two components: the *routing* module and the *disambiguation* module. The routing module takes a caller request and determines a set of destinations to which the call can reasonably be routed. If there is exactly one such destination, the call is routed there and the customer notified; if there are multiple destinations, the disambiguation module is invoked in an attempt to formulate a query; and if there is no appropriate destination or if a reasonable disambiguation query cannot be generated, the call is routed to an operator. Figure 1 shows a diagram outlining this process.

The focus of this paper is on the routing module, which begins with **term extraction.**. Given a transcription of the caller's utterance (either from a keyboard interface or from the output of a speech recognizer), the first step is to extract the relevant n-gram terms from the utterance. For instance, term extraction on the request *"I want to check the balance in my savings account"* would result in one bigram term, *"saving,account"*, and two unigrams, *"check"* and *"balance"*.

The next step in routing is **pseudo-document generation**. Given the extracted terms from a caller request, we can represent the request as an m-dimensional vector Q where each component

[1] These 3753 calls are a subset of the corpus of 4497 calls used in our corpus analysis. We excluded those ambiguous calls that were not resolved by the operator.

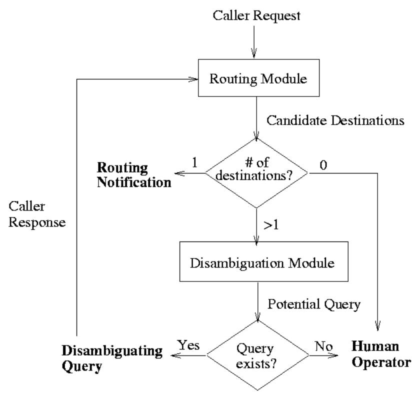

Figure 1: Call Router Architecture

Q_i represents the number of times that the ith term occurred in the caller's request. We then create an r-dimensional *pseudo-document* vector $D = QU$, following standard methodology (see Deerwester et al. 1990). Note that D is simply the sum of the term vectors U_i for all terms occurring in the caller's request, weighted by their frequency of occurrence in the request, and is scaled properly for document/document comparison.

Next, we perform **scoring**. Once the vector D for the pseudo-document is determined, we compare it with the document vectors by computing the cosine between D and each scaled document vectors in VS. Next, we transform the cosine score for each destination using a sigmoid function specifically fitted for that destination to obtain a confidence score that represents the router's confidence that the call should be routed to that destination.

The reason for the mapping from cosine scores to confidence scores is because the absolute degree of similarity between a request and a destination, as given by the cosine value between their vector representations, does not translate directly into the likelihood for correct routing. Instead, some destinations may require a higher cosine value, i.e., a closer degree of similarity, than others in order for a request to be correctly associated with those destinations. Thus we collected, for each destination, a set of cosine value/routing value pairs over all calls in the training data, where the routing value is 1 if the call should be routed to that destination and 0 otherwise. Then for each destination, we used the least squared error method in fitting a sigmoid function, $1/(1+e^{-(ax+b)})$, to the set of cosine/routing pairs.

We tested the routing performance using cosine vs. confidence values on 307 unseen unambiguous requests. In each case, we selected the destination with the highest cosine/confidence score to be the target destination. Using strict cosine scores, 92.2% of the calls are routed to the correct destination. On the other hand, using sigmoid confidence fitting, 93.5% of the calls are correctly routed. This yields a relative reduction in error rate of 16.7%.

The fourth step in the routing process involves **deciding** what to do with the scoring results. The outcome of the routing module is a set of destinations whose confidence scores are above a pre-determined threshold. These candidate destinations represent those to which the caller's request can reasonably be routed. If there is only one such destination, then the call is routed and the caller notified; if there are two or more possible destinations, the disambiguation module is invoked in an attempt to formulate a query; otherwise, the the call is routed to an operator.

To determine the optimal value for the threshold, we ran a series of experiments to compute the upperbound and lowerbound of the router's performance varying the threshold from 0 to 0.9 at 0.1 intervals. The lowerbound represents the percentage of calls that are routed correctly, while the upperbound indicates the percentage of calls that have the potential to be routed correctly after disambiguation (see section 6. for details on upperbound and lowerbound measures). The results in Figure 2 show 0.2 to be the threshold that yields optimal performance.

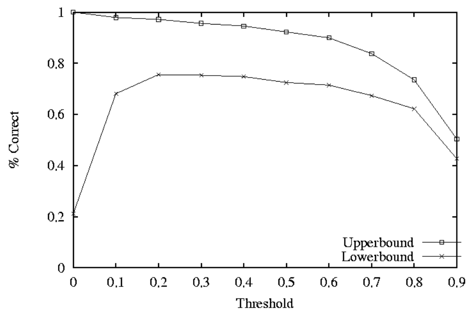

Figure 2: Router Performance vs. Threshold

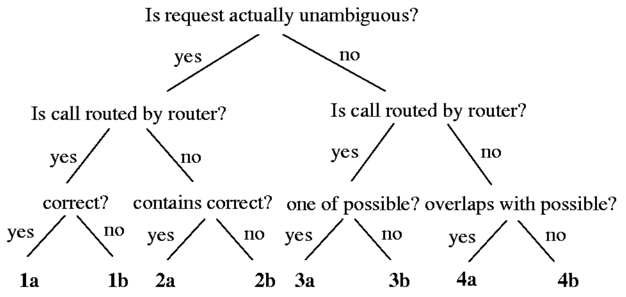

Figure 3: Classification of Router Outcome

Our system employs the same vector-based representations of terms and documents in order to generate clarification questions in order to **disambiguate** vague or ambiguous user queries. This process is described and evaluated in (Chu-Carroll and Carpenter 1998).

6. EVALUATION

We performed an evaluation of the routing module of our call router on a fresh set of 389 calls disjoint from the training corpus. Of the 389 requests, 307 were unambiguous and routed to their correct destinations, and 82 were ambiguous and annotated with a list of candidate destinations. Unfortunately, in this test set, only the caller's first utterance was transcribed. Thus we have no information about where the ambiguous calls should be routed after disambiguation.

The routing decision made for each call is classified into one of 8 groups, as shown in Figure 3. For instance, group **1a** contains those calls which are actually unambiguous, are considered unambiguous by the router, and are routed to the correct destination. On the other hand, group **3b** contains those calls which are actually ambiguous, are considered by the router to be unambiguous, and are routed to a destination which is not one of the potential destinations.

We evaluated the router's performance on three subsets of our test data: unambiguous requests alone, ambiguous requests alone, and

	Unambiguous Requests	Ambiguous Requests	All Requests
LB	1a/(1+2)	4a/(3+4)	(1a+4a)/all
UB	(1a+2a)/(1+2)	(3a+4a)/(3+4)	(1a+2a+3a+4a)/all

Table 3: Calculation of Upperbounds and Lowerbounds

	Unambiguous Requests	Ambiguous Requests	All Requests
LB	80.1%	58.5%	75.6%
UB	96.7%	98.8%	97.2%

Table 4: Transcription Results, no rejection, threshold = 0.2

all requests combined. For each set of data, we calculated a lowerbound performance, which measures the percentage of calls that are correctly routed, and an upperbound performance, which measures the percentage of calls that are either correctly routed or have the potential to be correctly routed. Table 3 shows how the upperbounds and lowerbounds are computed based on the classification in Figure 3 for each of the three data sets. For instance, for unambiguous requests (classes 1 and 2), the lowerbound is the number of calls actually routed to the correct destination (1a) divided by the number of total unambiguous requests, while the upperbound is the number of calls actually routed to the correct destination (1a) plus the number of calls which the router finds to be ambiguous between the correct destination and some other destination(s) (2a), divided by the number of unambiguous queries. The calls in category 2a are considered to be potentially correct because it is likely that the call will be routed to the correct destination after disambiguation.

Table 4 shows the upperbound and lowerbound performance for each of the three test sets on the transcribed text of the callers' utterances. These results show that the system's overall performance in the case of perfect recognition will fall somewhere between 75.6% and 97.2%. The actual performance of the system is determined by two factors: 1) the performance of the disambiguation module, which determines the correct routing rate of the 16.6% of the unambiguous calls that were considered ambiguous by the router (class 2a), and 2) the percentage of calls that were routed correctly out of the 40.4% ambiguous calls that were considered unambiguous and routed by the router (class 3a). Note that the performance figures in Table 4 are the result of 100% automatic routing, since no request in our test set failed to evoke at least one candidate destination. In (Chu-Carroll and Carpenter 1998), we evaluate the performance of the disambiguation module, which determines the overall system performance, and show how allowing calls to be rejected and punted to operators affects the system's performance.

We also evaluated our system on the output of a large vocabulary, speaker independent, continuous speech recognition system. The recognizer is described in detail in (Reichl et al. 1998). Here we simply repeat the precision and recall scores for content term extraction, because these are the only terms whose correct recognition affects routing performance. Our unigram precision/ recall was 94.1%/87.9%, our bigram precision/recall was 96.9%/85.4% and our trigram precision/recall was 98.5%/84.3%. Thus our recognizer completely misses around 12% of all relevant content terms, but is quite reliable when it hypothesizes a content term, especially a bigram or trigram.

One potential effect of missing n-gram terms is that the call router no longer has sufficient information to route unambiguous calls. Furthermore, a possible effect of extra n-gram terms is that they may be added to an otherwise unambiguous set of terms, causing uncertainty in routing. We tested this hypothesis by comparing the number of calls that are considered ambiguous by the router when using transcription versus ASR results as input. On our test set, we found that the number of calls classified as ambiguous rises from 26.7% on transcriptions to 27.8% on ASR output.

	Unambiguous Requests	Ambiguous Requests	All Requests
LB	77.9%	47.6%	71.5%
UB	90.6%	86.6%	89.7%

Table 5: ASR Results, no rejection, threshold = 0.2

In Table 5, we provide an evaluation on ASR output in the same format as the transcription results in Table 4, using the same threshold of 0.2 and no rejection. We show, in (Chu-Carroll and Carpenter 1998), that the effect of rejecting 10% of the calls to a human operator significantly increases performance on transcribed calls; similar effects hold for queries processed by speech recognition.

7. CONCLUSION

Our primary conclusion is that it is time to replace touch-tone menu systems with spoken language understanding systems.

We described and evaluated a domain independent, automatically trained call router that is able to route calls and engage in disambiguation dialogues. We have demonstrated that the system is robust in the face of noise introduced by ASR: with no rejection, the lowerbound on performance drops from 75.6% to 71.5%, and the upperbound (after disambiguation) drops from to 97.2% to 89.7%.

REFERENCES

Chu-Carroll, J. and B. Carpenter. 1998. Dialogue management in vector-based call routing. In *ACL/COLING '98*, 256–262.

Deerwester, S., S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Gorin, A., G. Riccardi, and J. Wright. 1997. How may I help you? *Speech Communication*, 23:113-127.

McDonough, J., K. Ng, P. Jeanrenaud, H. Gish, and J. R. Rohlicek. 1994. Approaches to topic identification on the switchboard corpus. In *ICASSP '94*, 385–388.

Reichl, W., B. Carpenter, J. Chu-Carroll, W. Chou. 1998. Language modeling for content extraction in human-computer dialogues. In *ICSLP '98*. Sydney.

Salton, G. 1971. *The SMART Retrieval System*. Prentice Hall.

Schütze, H., D. Hull, and J. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. In *SIGIR '95*.

Sparck Jones, K. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–20.

Sproat, R., editor. 1998. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Kluwer.