4 Confidence Confirmation Algorithm in Automatic Speech Recognition System

Over the recent years, speech recognition technology has been making steady and significant progress. Advances in natural language generation algorithms, robust parsing techniques, and the advent of high-quality speech synthesis systems find the way for the emergence of robust speech recognition systems.

This research study has focused on the process involved in finding out what the user has intended in each of his dialogue turn in a mixed dialogue, trained or accustomed on a speech recognizer word graph with associated word confidence measures.

The dialogue component of the system directly influences the selection process of the initial stage when it has provided with a particular context. And the dialogue manager is the the component which has local information from every server and also global knowledge of a particular user’s various constraints.

Here, in this study, Confidence Confirmation Algorithm (CCA) in the selection of recognition hypothesis in the framework of machine-human interaction is described. Enhancement has been made to other human language technology servers which serve the purpose in providing the necessary information to the dialogue manager and new capabilities in the dialogue manager is intended at repairing and detecting problematic and error spots in the dialogue.
4.1 Introduction

SRS generally order the hypotheses by computing scores for each of the utterance hypotheses. The scores produced by the hypothesis are useful for ordering the hypotheses based on preference, but do not give a good indication of the quality of the recognition or how much confident the system is that the decoding is correct.

For applications to act on speech input, the applications must be able to assess the confidence that the input has been decoded in a proper manner. This work extended and combined the work described in [?] and is related to extending another feature [?] for providing confidence annotation for the speech recognizer output. The idea is to normalize decoded word strings and the phone acoustic scores produced by a less constrained search.

[?] used an all-phone recognition to normalize the scores of the hypotheses, followed by Bayesian updating. Among other things, [?] also used the best matching observation for every frame(senone) to normalize the acoustic score for the hypothesis. For acoustic score measure, 10ms frame-level observation scores as the basis for the normalization is used.

Sphinx-4 system [?] is used specifically as speech recognizer for this study. It is a Semi-Continuous HMM recognizer using a trigram language model. The acoustic observations are modeled in this system by senones [?]. Senones are the tied HMM state mixture weights especially for the Gaussian distribution used by the semi-
In speech recognition system, one of the most difficult aspects is to make sure that the system clearly and correctly understood each user’s query. Only then the system must be able to recover efficiently and gracefully from errors.

A tedious but effective strategy is to prompt the user in every turn, giving only one piece of information, immediately verifying through a confirmation sub-dialogue whether it has been correctly understood. A more natural interface would allow the user much more freedom, but at the cost of significantly higher perplexity.

In a mixed-initiative system, it becomes generally important to draw as many constraints as possible to aid in the selection task of the hypothesis. Naturally explicit confirmation of the dialogue can yield better confidence in the validity of hypothesized utterances, but there is again the risk of increased tediousness.

The problem based on utterance-level confidence measure \[ ? \] can generally be considered as a machine learning classification task. That is, when the user utterance the given a set of relevant features is selected and is used to classify the utterance as whether correctly understood or not.

This study discusses how the Airline database deals with the issues of hypothesis selection and verification. A mixed initiative dialogue strategy is utilized and supported by confirmation sub-dialogues that are invoked only when the system actively suspects any miscommunication.

The system poses challenging and interesting problems for dialogue systems in that
the interaction is complex and involves several variables. Suppose these variables are specified, users may become quite confused and the dialogue can be derailed if a serious mis-recognition occurs.

This study also describes about both the hypothesis selection process and the method that is used to control dialogue management. Finally, this study describes about confidence confirmation algorithm (CCA), which as a policy confirms when it detects an unexpected response from the user.

4.2 System Description

The system performs a number of domain-specific functions and applications like spoken dialog system interfaces. The functions include retrieval of information on the web, access of information in the system database and domain-specific reasoning. The interfaces in web-based resources are used to obtain information about the various flights. Other information includes trip schedules, cost of tickets for flights and locations.

The system combines domain-specific reasoning for example, resolving ambiguous references and managing solution sets like ranking flights based on user’s interest. This system interacts with the database, that contains the geographical information (about 500 destinations) and information about airlines.

The database contains the information of how the users might refer to various information in the domain. For example airport names and information about how

Published in Journal of Theoretical and Applied Information Technology, 39(1), 2012, ISSN: 1817-3195
the system should in turn refer entities while speaking to the user.

4.2.1 Base Architecture

Speech recognition is written entirely in the programming language of Java. The architecture is based on Sphinx primarily developed at CMU (www.speech.cs.cmu.edu/sphinx). Goals of Sphinx-4 speech recognition engine have

- Highly flexible recognizer

- Performance of this speech recognizer is equal to or exceeds the performance of Sphinx-3

- Collaborated with researchers at MERL, and CMU and Sun Microsystems and others.

The architecture used for the example of the travel information system is Sphinx-4 architecture. The main components of Sphinx-4, combine together during recognition process. When the recognizer starts up, it constructs the front end which ultimately generates features from speech input, the decoder, and the linguist generates the required search graph according to the configuration specified by the user.

These components are helpful in constructing their own subcomponents. For example, the linguist will construct the dictionary, language model and the acoustic model. It uses the knowledge from these three major components to construct a search
graph which is very appropriate for the recognition task. Decoder will construct the search manager, which in turn constructs the pruner, the scorer and the active list.

**Sphinx-4 Loader**

The acoustic model generated by Sphinx-4 trainer is loaded by the loader. A jar file contains the package of the acoustic model. The package need not contain the language and dictionary model files. The data file which make up the acoustic model and the text model are very much needed. The text file contains key-value pairs which are loaded by the loader. The text files are likely to have properties like

- Location of the data- contains the directory where the actual data files are stored.

- Definition of the model- locate the model definition file.

**FSTGrammar**

Thid FST grammar loads the grammar from a file which represents a finite-state transducer (FST) in the grammar format of *'ARPA'*'. Probabilities from the FST file are always in negative log format and are converted to internal log base.

When the FST file is read, a structurally equivalent Grammar object equivalent to FST is created. The steps of changing the FST file into Grammar object are namely

- Creation of Grammar nodes - For the entire FST file and word transition, the destination node ID is taken grammar node using ID is produced. This is placed

Published in Journal of Theoretical and Applied Information Technology, 39(1), 2012, ISSN: 1817-3195
in hash table to be sure that they are created for every ID. Hence, one word for each grammar node is created.

- Creation of end node for every Grammar node - The end node is mainly used for the back off transition to form the Grammar node and so it will not go to the word itself, but will go straight to end of the word. Optional silence node between the end node and its grammar node is added.

- Creating the transition - The entire FST file is read, and each line indicating a transition is connected with the corresponding Grammar nodes. Null transitions and Back-off transitions are linked to the end node of the grammar node.

**Dialogue Manager**

The Dialogue manager is a component which has not only local information from every server, but it has a global knowledge about a unique users constraints.

- The dialogue manager had the responsibility to determine how best it could answer to every users query

- It processes the users query

- It is represented as a semantic frame

- Prepares its meaning response and represent the response as a semantic frame
• The generation component then converts the reply frame into well formed reply string and is spoken back to user.

• The dialogue control is generally managed with the help of a dialogue control table.

4.3 Confidence Confirmation Algorithm (CCA)

Confidence Confirmation Algorithm (CCA) is mainly concerned with the hypothesis selection process which is a complex process that involves several steps, which includes interactions among multiple servers. This process is represented schematically in the algorithm given below.

Confidence Confirmation Algorithm

Step 1: Find initially the word graph representing multiple sentence hypothesis along with the associated confidence scores for every word in the graph.

Step 2: Construct an n-best list of semantic frames, including alternative candidates for the meaning of the utterance.

Step 3: Find the most promising of among all the frames, consider possible discourse context and then presents this candidate to the dialogue manager.

Step 4: The dialogue manager finally decides whether this request is consistent with the earlier dialogue. When some part of the query becomes problematic the things
that could be possible are

(i) Ask for explicit confirmation from the user

(ii) Seek an pragmatically appropriate alternative hypothesis from the n-best list

(iii) delete certain attributes which are both pragmatically inappropriate and low scoring

(iv) Initiate a sub-dialogue if needed asking for confirmation

(v) Request the user to keypad in the information

**Step 5:** Send the information to the recognizer again to get the confidence scores for every word.

**Step 6:** Stop

The speech recognizer processes the recorded waveform of the user and generates a word graph with associated confidence scores for each word in the graph [?]. The confidence scores are mainly based on the log likelihood probabilities of the corresponding words which are obtained from the acoustic models for their component phones.

The confidence scores are obtained from a possible set of features which are combined to produce a single score using linear discriminant techniques [?]. In addition to the minimum and mean log likelihood score of the word in all of its possible local alignments, the combined confidence score takes into account the difference between
the word’s score and the its best score obtainable over the same acoustic space, and against the score of a “catch-all” model \([?)\]. Here the number of competitors for the acoustic region is also taken into account.

![Figure 4.1: Hypothesis selection and verification](image)

The Major step in hypothesis selection is to parse the speech recognizer’s word graph into a possible set of candidate semantic frames. It is done with the help of the natural language system which parses from a context free grammar combined with feature unification and a trace mechanism for movement. Linguistic and Acoustic scores are combined to give the score of the overall sentence.

In addition to the combined total score for each hypothesis, critical content words(e.g., dates and cities) contain their confidence score associated with the corresponding element in the semantic frame, for the possible consideration by the dialogue manager later.
Every candidate semantic frame is labeled according to its parse status, with the maximum of four expected categories namely "robust parse", "phrase parse", "full parse" and "no parse". "Full parse" means that a single coherent parse tree for every word in the hypothesis. "Robust parse" means that every word was considered for, but the parse structure contains a sequence of parsed fragments with possibly interspersed licensed "skip words".

Here "Phrase spot" means that major parts of the hypothesis may be totally ignored, but a certain critical, high scoring, content words were singled or taken out for parsing. Even with all of these back-off mechanism, it is not always possible to parse some user utterances. Here, the dialogue manager is responsible for providing a context-dependent response for the "no parse" category.

The next procedure is to make use a simple heuristic to select the most promising candidate from the set of parsed frames. In case of absence of any directives from the dialogue component, the system ready to choose the highest scoring, backing off to robust-parse, and finally phrase-spotting. Even then, it always happen that the dialogue component has set up context conditions that will preferably favor an otherwise sub-optimal theory.

This is included with a list of one or more semantic categories which are in focus and in some cases, individual words which are highlighted, or individual words which are to be selected against. If the system has just asked the user for a return date, then all dates are given treatment preferentially. Similarly, if it has just listed the cities it knows
in a particular place, those particular cities may be highlighted.

After finding the most promising hypothesis has been singled out, it is processed through context resolution component and delivered to the dialogue manager for further consideration. If everything goes well and the new information is interpreted, a response is prepared that moves the dialogue plan closer to the conclusion. Alternate hypotheses are retained, but utilized only when there is a reason to believe the selected hypothesis contain errors.

**Dialogue Control**

The dialogue manager is tasked with the responsibility of determining how best it is possible to answer each user’s query. In each turn, it processes the user’s query, represented as a meaningful semantic frame, and prepares its meaning response and finally represent as a semantic frame.

The generation component converts this reply frame into a well formed reply string, to be spoken back again to the user. The dialogue control component is managed through the use of a dialogue control table. Fig. 4.1 takes the form of a set of rules, specifying the functions to be called when specified conditions are met.

The conditions are tests (string match, arithmetic, boolean etc.) on variables maintained in a dynamic dialogue state frame. Fig. 4.2 depicts a simple dialogue control for managing complexity. It plans for a linear organization of the complex planning tasks of dialogue management that provides a high-level representation of
dialogue activities in an outline form.

The variables are initialized from the user’s query that is based on the context, and are augmented in the course of a dialogue turn by the various functions that are executed. Then the system developer partition the dialogue tasks into a set of specific functions, and choreograph the order in which, and conditions under which every function is called.

Ideally, every function has a very specific role for it, some having to verify that the query is fully specified, some others involved with retrieving the information from the database and still others involved with preparing the reply frame. A selected subset of the rules concerned with managing dates is shown as in Fig. 4.1.

4.4 Specific Knowledge Sources

To inform hypothesis selection at any point in the dialogue, many knowledge sources are updated and maintained continuously throughout the user’s conversation with the system. The dialogue state is, of course gives the most useful of these knowledge sources. Then the dialogue state encode parts of either side of the conversation, in which it could identify any preceding system initiated query as well as user-specified constraints.

The dialogue state contains information on how far the user has arrived in the overall travel plan, which are helpful in determining whether a particular dialogue move is likely. This system also retains a history of user model which is continually augmented.
as the dialogue progresses through the itinerary plan.

It includes many as yet enforceable, such as early specification of return date or mention the desired fare class before the itinerary is completed. This also includes the very particular details of the selected partial itinerary, which are particularly useful for applying date and source constraints to later legs.

In addition to the earlier ones a set of frames are maintained for alternative recognizer hypotheses for dates and for possible later reselection. If the original date is pragmatically incredible, missing data recognition alternative dates are selected from the set. Then the alternatives are flushed each time a new one is selected.

A thorough record-keeping mechanism is done for destination and tracking source throughout a dialogue. Two keys are especially problematic, particularly in cases where the user may be attempting to travel to a city or from a city that is not in the recognizer’s vocabulary.

Like these cases, the same mis-recognition tends to happen repeatedly, because the recognizer tries to substitute the same incorrect hypothesis for the intended city, or the source/destination in question varies from query to query, because the recognizer hypothesizes different cities in its known vocabulary. Monitoring the patterns of source/destination keys from query to query, it could be possible to decide whether to prompt for verification or to solicit keypad input.

Here in Fig. 4.2, the source is inherited in the last two utterances, whereas the destination is first repeated and then changed. Note: MAA to represent Chennai
{ city_history
  :source “MAA” :source_status “inherited”
  :dest “BOM” :dest_status “changed”
  :history { city_history
    :source “MAA” :source_status “inherited”
    :dest “BOM” :dest_status “repeated”
    :history { city_history
      :source “MAA”
      :source_status “first”
      :dest “BOM”
      :dest_status ”first” } } }

Figure 4.2: Example frame to represent city history.

International Airport and BOM to represent Mumbai International Airport.

Each source and destination city is entered into this history throughout the course of a single dialogue. This history is updated for each turn in which these values are present, either from the user utterance or from inheritance. Table 4.1 gives a dialogue with the user involving logging on.

Table 4.1: A dialogue with the user involving logging on

<table>
<thead>
<tr>
<th>S:</th>
<th>Welcome...Please login saying your login name. Else say ”New user”</th>
</tr>
</thead>
<tbody>
<tr>
<td>U:</td>
<td>Kavitha</td>
</tr>
<tr>
<td>S:</td>
<td>please say the password...</td>
</tr>
<tr>
<td>U:</td>
<td>August first[misrecognized]</td>
</tr>
<tr>
<td>S:</td>
<td>Sorry I didn’t catch that. Please enter your password using soft keyboard</td>
</tr>
<tr>
<td>U:</td>
<td>[successful enrollment]</td>
</tr>
</tbody>
</table>

A status is stored along with the city, indicating whether the city was newly

Published in Journal of Theoretical and Applied Information Technology, 39(1), 2012, ISSN: 1817-3195
introduced in that turn, changed, repeated, or inherited from a previous turn. The record is stored in a nested frame structure, as illustrated in Fig. 4.2. A sample dialogue for air reservation is given as in Fig. 4.3.

Here's a sample run:

Welcome to the Sphinx-4 Dialog

Loading dialogs
Loading the recognizer
Running

======= menu =======
ext the program
get help
trip diary
hot news
I shows listing of flights
listen to music
weather forcaster

======= menu =======
ext the program
get help
listen to airline news
make a phone call
read my email
show statistics
weather forcaster

Figure 4.3: A sample dialogue for air reservation

The dialogue is given where the keypad entry was successful for enrolling the password during the logging on stage which is depicted in Table 4.1. For every query, containing source or destination keys, the record is consulted to determine if the values are consistent with what was appeared before in the dialogue. The city history is flushed
wherever the flight is selected for the new itinerary.

4.4.1 Conclusion

In SRS, although each separate server can be trained and developed on its own, the system function as part of the entire system, and this happen in the context of a dialogue with a human user. The dialogue manager is the one component that has not only local information from each server, but also global knowledge about a particular user’s constraints.

In this research study, CCA is described in the selection of recognition hypotheses in the context of human-machine interaction. New capabilities in the dialogue manager aimed at detecting and repairing problematic spots in the dialogue are described here. The dialogue manager component directly influences the initial selection process, at least whenever it has provided a specific context. While a set of n-best semantic frame is produced, most of the attention is directed towards the primary selected candidate.

After careful perusal, several problematic situations trigger a response that involves confirmation requests and/or help messages. Even Sometimes components of the frame are ignored, either because the system can find no appropriate interpretation for them, they have low confidence scores, and/or they conflict with other information present in the same frame.

Thus the general strategy is to invoke confirmation sub-dialogues only when the user appears to make a surprise move. Similarly, alternative hypotheses are only
considered when the top hypothesis leads to pragmatically implausible outcomes.