Machine Learning for Recommendation System

Synopsis of the Thesis to be submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

by

Souvik Debnath
(Roll No: 06CS6036)

Under the supervision of

Dr. Pabitra Mitra and Dr. Niloy Ganguly

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur. May 2008

Contents

Abstract

Recommendation system has been seen to be very useful for user to select an item amongst many. Most existing recommendation systems rely either on a collaborative approach or a content-based approach to make recommendations. We have applied machine learning techniques to build recommender systems. We have taken two approaches. In the first approach a content based recommender system is built, which uses collaborative data, so that it gets the effect of a hybrid approach to get better result of recommendation. Attributes used for content based recommendations are assigned weights depending on their importance to users. The weight values are estimated from a set of linear regression equations obtained from a social network graph which captures human judgment about similarity of items. In the second approach agent of call centre have been recommended some procedure depending on the current state of online call. A combination of K-Means Algorithm and Hidden Markov model is used.

1 INTRODUCTION

For many years recommendation systems had been a part of many online shopping systems. But in recent years it is evolving as a part of many other systems like portals, search engines, blogs, news, WebPages etc. We can put recommendation system on a top of another system, which have mainly two elements Item and User. To build the recommendation system one can use the Item data of the underlying system or both Item and User data. Examples of items are book, song, movie, news, blog, procedure etc.

There are mainly two approaches to build a recommendation system- *Collaborative Filtering (CF)* or *Social Information Filtering (SF)* and *Content Based (CB)*. Collaborative Filtering system maintains a database of many users' ratings of a variety of items. For a given user, it finds other similar users whose ratings strongly correlate with the current user. It recommends items which are rated highly by these similar users, but not rated by the current user. Almost all existing commercial recommenders use this approach (e.g. Amazon). To build a Collaborative Filtering system one need to use both user and item data. But Content Based system uses only the item data. It maintains a profile for each item. Considering the attributes or feature of the item it CB finds the similarity between items, and recommends the most similar item for an item.

We have worked on two applications of content based recommendation systems. The first one is movie recommendation to the user. In this application when a user hits or selects one movie, or opens a page of a movie, the recommendation system recommends other movies which are similar to that selected movie. Similarity measurement has been done comparing the feature set of movies. The second application we have considered is call center, where the agents are recommended procedures that they should follow depending on the present status of the call. When a customer calls to the call center, while getting the query from the customer the agent access some knowledge base for the possible solution or answer to the customer's query. Instead of this manual access to the knowledge base, prompting the agent some recommendation of possible solution would be very effective. Depending on the current content of the call, it will produce a list of possible solution automatically.

2 MOVIE RECOMMENDATION USING SOCIAL NETWORK ANALYSIS

2.1 Related Work and Motivation

Collaborative filtering computes similarity between two users based on their rating profile, and recommends items which are highly rated by similar users. However, quality of collaborative filtering suffers in case of sparse preference databases. Content based system on the other hand does not use any preference data and provides recommendation directly based on similarity of items. Similarity is computed based on item attributes using appropriate distance measures. We attempt to hybridize collaborative filtering and content based recommendation for circumventing the difficulties of these individual approaches. Item similarity measure used in content based recommendation is learned from a collaborative social network of users.

Some previous attempts at integrating collaborative filtering and content based approach include content boosted collaborative filtering [3], weighted, mixed, switching and feature combination of different types of recommender system [2]. But none of these talks about producing recommendation to a user without getting her preferences. We demonstrate the effectiveness of the proposed system for recommending movies in Internet Movie Database (IMDB) [1]. From the results it is seen that our recommendation is quite in agreement with IMDB recommendation.

2.2 Algorithm

In content based recommendation every item is represented by a feature vector or an attribute profile. The features hold numeric or nominal values representing certain aspects of the item like color, price etc. A variety of distance measures between the feature vectors may be used to compute the similarity of two items. The similarity values are then used to obtain a ranked list of recommended items. If one considers Euclidian or cosine similarity; implicitly equal importance is asserted on all features. However, human judgment of similarity between two items often gives different weights to different attributes. For example, while choosing a camera, price of a camera may be more important than the body color attribute. It may be stated that users base their judgments on some latent criteria which is a weighted linear combination of the differences in individual attribute. Accordingly, we define similarity S between objects O_i and O_j as

$$S(O_i, O_j) = \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \cdots + \omega_n f(A_{ni}, A_{nj}) \tag{1}$$

where ω_n is the weight given to the difference in value of attribute A_n between objects O_i and O_j, the difference given by $f(A_{ni}, A_{nj})$. The definition of $f()$ depends on the type of attribute (numeric, nominal, Boolean). We normalize f's to have value in [0, 1]. In general the weights $\omega_1, \omega_2, \cdots, \omega_n$ are unknown. In the next section we describe a method of determining these weights from a social collaborative network.

We have used the above methodology for recommending movie in IMDB database. A set of 13 features are considered. The features along with their type, domain and distance measures are shown in Table 1. All these feature values can be obtained from the IMDB database.

We estimate the feature weights from a social network graph of items. The underlying principle is to use existing recommendation by users to construct a social network graph with items as nodes. The graph represents human judgment of similarity between items aggregated over a large population of users. Optimal feature weights are considered to be those which induce a similarity measure between items best conforming to this social network graph.

Table 1: Features Used in Movie Recommendation

Feature	Type	Domain	Distance Measure
Release	Year	YYYY	$\frac{(300-\|Y_1-Y_2\|)}{300}$
Type	String	Movie,TV etc.	$T_1 = T_2?1:0$
Rating	Integer	(0-10)	$\frac{(10-\|R_1-R_2\|)}{10}$
Vote	Integer	(≥ 5)	$\frac{(V_{max}-\|V_1-V_2\|)}{V_{max}}$
Director	String	<Name>	$D_1 = D_2?1:0$
Writer	String	<Name>	$W_1 = W_2?1:0$
Genre	(String)*	Drama etc.	$\frac{\|G_1 \cap G_2\|}{G_{max}}$
Keyword	(String)*	College etc.	$\frac{\|K_1 \cap K_2\|}{K_{max}}$
Cast	(String)*	(<Name>)*	$\frac{\|C_1 \cap C_2\|}{C_{max}}$
Country	(String)*	France etc.	$\frac{\|C_1 \cap C_2\|}{C_{max}}$
Language	(String)*	English etc.	$\frac{\|L_1 \cap L_2\|}{L_{max}}$
Color	String	Color, B/W	$C_1 = C_2?1:0$
Company	String	<Name>	$C_1 = C_2?1:0$

We describe below a linear regression framework for determining the optimal feature weights. Let the items under consideration be denoted by $O_1, O_2, \cdots, O_l$, they corresponds to the vertices of our social network. The edge weight between vertices O_i and O_j,

$E(O_i, O_j) =$ # of users who are interested in both O_i, O_j.

$E(O_i, O_j)$, suitably normalized, may be considered as human judgment of similarity between O_i, O_j. Recall that feature vector (content based) similarity between O_i, O_j has been defined as $S(O_i, O_j)$ in Eq. (1). Equating $E(O_i, O_j)$ with $S(O_i, O_j)$ leads to the following set of regression equations. $\forall i, \forall j = 1..l \wedge i \neq j$,

$$\omega_0 + \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \cdots + \omega_n f(A_{ni}, A_{nj}) = E(O_i, O_j) \tag{2}$$

The values of $f(A_{1i}, A_{1j}), f(A_{2i}, A_{2j}), \cdots, f(A_{ni}, A_{nj})$ are known from the data as are the values of $E(O_i, O_j)$. Solving the above regression equations provide estimates for the values of $\omega_1, \omega_2, \cdots, \omega_n$. If there are l objects under consideration, it is possible to have ${}^lC_2$ regression equations of the above form. In the case of movie recommendation we have considered movies as nodes in the social network. The edge weight between two movies is the number of IMDB reviewers who have reviewed both the movies.

2.3 Experimental Results

The movie database used in our recommendation system consists of 3×10^5 random movies downloaded from the IMDB. The movies voted by less than 5 people or the movies that have not been reviewed by a single person are filtered out. The data is then divided into three equal sets. Each movie is described by 13 features (Table 1).

2.3.1 Stability of Feature Weights

Our recommendation system is based on the presumption that feature weights are almost universal for different sets of users and movies. To test this presumption we consider different sets of regression equations and solve for the weights. We consider the following varieties of regression equations.

I. Equations using only edge weights ≥ 1 (i.e. movie pairs having at least one co-reviewer)

II. Equations using only edge weights ≥ 2 . (Note that this gives a graph which is a sub-graph of the previous graph.)

For the above graphs we construct a set of equations for each of the three (partitioned) datasets having 10^5 movies. Thus we get six sets of regression equations which we solve using SPSS package. It is observed from the weight values obtained from each of the above six sets of regression equations that some of the features have stable weight values, while some features like Director, Rating, Vote, Year, Color have unstable or negative weight. We remove the features with unstable or negative weights from our regression equations and obtain the following set (Table 2) of stable weights for eight features. Also note, out of the 8, 3 features namely type, writer and company are particularly important. These features along with their weights are used to obtain the recommendations.

Table 2: Feature Weight Values

Feature	Mean	Variance
Type	0.18	0.0023
Writer	0.36	0.0048
Genre	0.04	0.0001
Keyword	0.03	0.0011
Cast	0.01	0.0003
Country	0.07	0.0013
Language	0.09	0.0004
Company	0.21	0.0110

2.3.2 Performance of the Recommender System

The proposed algorithm is compared with pure content based method (considering equal weights for all features) and IMDB recommendations. Performance is measured using the classical *Recall* measure, considering IMDB recommendation as benchmark. The experiment has been done on 10 different movies. The proposed method achieves an average recall of 0.29. Where as, the pure content based method achieves a recall of 0.24 with IMDB. Thus the proposed method agrees well with IMDB recommendation and in this regard it outperforms pure content based method. This demonstrates the effectiveness of feature weighting.

3 PROCEDURE RECOMMENDATION TO CALL CENTER AGENT

3.1 Related Work and Motivation

Contact center (or *call center*) services are very common for various business models starting from product sell to handling customer issues. Contact center are mostly based on telephonic call. Some supports are provided by web-chat or email also. When a contact center agent get a call (or query), she tries to find the possible solution by searching the knowledge base. But that is very time consuming. So when ever a call comes to the agent, after some time depending on the current status of call the possible solution should be prompted to the agent automatically, so that the agent can respond to the call effitiently. There were some previous attempts on call center dialogs mining [4], [5]. But none of these considers affect of the sequence. We try to capture the sequence information and use it for clustering using HMM. In this work, we consider any conversational text derived from web-chat systems, voice recognition systems etc., and propose a method to identify procedures that are embedded in the text. We discuss here how to use the identified procedures in knowledge authoring and agent prompting.

3.2 Algorithm

Calls can be considered as sequences of information exchanges between the caller and the responder. A procedure refers to a particular flow of directed conversation. As the data is large we have taken a two level approach. At the first level we consider the whole corpus together and then cluster it into smaller partitions. In this clustering a complete call is considered as an element or document. The clustering is done in a bag of word approach. This gives a topic wise cluster. A topic refers a domain. A contact center can handle calls for different domains. At the second level of our approach we consider the unit of information exchange (procedure sub-steps) and the sequence of such exchange. This is done for each topical cluster separately. For each such topical cluster of calls, set of agent and customer turns (sentences) are collected. In a conversation agent and customer put their sentences alternatively. In this document the word turn is used to refer the sentences that one party has put in a single turn. Hence a call is a sequence of turns, in which agent turn and customer turn comes alternatively. These turns are clustered using K-Mean algorithm. Once SPTS clusters are obtained, each call can be represented as a sequence of SPTS clusters. To impose the effect of sequence into the clustering, Hidden Markov Model and Viterbi Algorithm are used. The STPS label sequence, which are found after KMA, are used to learn the HMM. In HMM, STPS labels are considered as hidden states. After learning the HMM, the turns are relabeled using Viterbi Algorithm. KMA, HMM parameter learning and Viterbi are done iteratively unless we get stability on some convergence criteria. Using the HMM a set of procedures have been generated considering the highest probability. Further, we evaluate the utility of these procedure collections in an agent prompting scenario and show its effectiveness over traditional techniques such as information retrieval on the same call corpus.

3.2.1 Finding Topical Clusters of Calls

As the data is huge two level of clustering is done. Typically very diverse issues are handled by call centers. So in the first level of clustering we abstract away the topical difference of the calls. This clustering has been done using the KMA algorithm and considering the whole call as a document which contains a concatenation of all the sentences in the call. The call has been represented as a vector of term frequencies. We set K to the number of different applications and issues that the concerned call center handles. Note that, it is good enough to make K equal to an approximate number rather than the exact number of applications or issues.

3.2.2 Obtaining Sub-Procedure Text Segment (SPTS) Clusters

Let C be the collection of calls $C_1, C_2, \cdots, C_N$. Each call C_i is represented by a sequence of turns $v_1(C_i), \cdots, v_{|C|}(C_i)$ where $|C|$ is the number of turns in the call. Each turn is associated with the speaker of that turn, which is from the set "Caller", "Responder". Let $Speaker(vi(Cj))$ be a function which returns the speaker of the i^{th} turn in call C_j. Let the length of the call C_i be n_i, i.e., $|C_i| = n_i \quad for \quad i = 1, \cdots, N$. Let $T_1, \cdots, T_k$ be a partition of C into K topic clusters. Let,

$$G_i = \bigcup_{\forall c \in T_i, \forall l, Speaker(v_l(c)) = "Caller"} v_l(c) \qquad (3)$$

be the set of sentences spoken by the caller in calls in T_i, and

$$H_i = \bigcup_{\forall c \in T_i, \forall l, Speaker(v_l(c)) = "Responder"} v_l(c) \tag{4}$$

be the set of sentences spoken by those who receive the calls in T_i. G_is and H_is are clustered separately to obtain SPTS clusters. We use the simple *K-Means algorithm (KMA)* to cluster the G_is and H_is. Considering the turn as a document, document clustering is done with a bag of word approach using KMA. Given a set of SPTSs and a call, the latter can be represented by a sequence of SPTSs with as many elements in the sequence as there are sentences in the call, and the i^{th} element of the sequence being that SPTS to which the i^{th} sentence in the call bears a maximum similarity with.

The above clustering does not consider the effect of the sequence of turns, rather it just consider the turn as a point in a vector space and cluster those points. A call can be considered as a stochastic process as a call is a sequence of turns where each turn is dependent directly on its previous turn and indirectly all other previous turns. In this figure Q_t is the turn at time instance t and Q_{t-1} is the previous turn and so on.

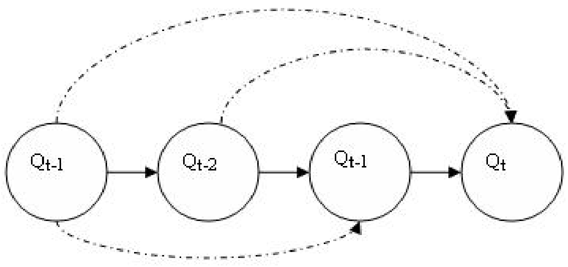

Figure 1: Turn Dependence

The arrow shows the dependency where dashed arrows are the indirect dependency. For Q_t, it is important to consider the edge from Q_{t-1} to Q_t but the edge Q_{t-2} to Q_t is not so important because the effect of that can be imposed with the edges Q_{t-2} to Q_{t-1} and Q_{t-1} to Q_t. So for simplicity we have considered only direct dependency. Hence, this model is similar to the first order Markov chain where probabilistic distribution of current state is only dependent on the predecessor state.

$$P[Q_t = Q_j | Q_{t-1} = Q_i, Q_{t-2} = Q_l,] = P[Q_t = Q_j | Q_{t-1} = Q_i] \tag{5}$$

If we consider the text of the turn as observation, our problem is to find the cluster (STPS label) of that turn. Hence we can fit a Hidden Markov Model here where the hidden state is the STPS label of the turn and the observed state is the turn text. To model it with HMM we need to define and learn the HMM parameters. We learn the HMM parameters and do the STPS clustering together with in an iterative fashion. Here is the sketch of algorithm that learn the HMM parameter and do the clustering together.

```
Corpus = Collection of calls of a specific topical cluster
Start with random centroids for KMA

While (converges){
        1. Do KMA
        2. Label each turn with cluster
```

```
    3. Learn HMM Param, $\gamma = (\pi, A, B)$
    4. Viterbi to re-label the turn with cluster
    5. Calculate the cluster centroids and use that as the start centroid for KMA of next iteration
}
```

3.2.3 HMM Parameter learning

An HMM is a double stochastic process in which there is an underlying stochastic process of hidden states and the observation state generated from the hidden state.
1) Stochastic process of hidden states $q_1, q_2, \cdots, q_t, \cdots, q_T$, where
t : discrete time, regularly spaced
T : length of the sequence
$q_t \in Q = q_1, q_2, \cdots, q_K$
K : the number of possible states
2) each state emits an observation according to a second stochastic process : $x_1, x_2, \cdots, x_t, \cdots, x_T$,where
$x_t \in X = x_1, x_2, \cdots, x_N$
x_i : a discrete symbol
N : number of symbols (Turns)

A complete specification of an HMM (γ) requires its three probability measure to be defined, transition probability (A), emission probability (B) and the initial probability (π). $\gamma = (\pi, A, B)$.

In our case initial STPS clustering gives a cluster label to all the turns. We represent a call as a sequence of states. From these sequences we calculate the transition, emission and initial probability.

Transition probability is the probability distribution of the transition between states.

$$P[q_t = q_j | q_{t-1} = q_i] = a_{ij} \qquad where \quad 1 \leq i, j \leq K \tag{6}$$

This defines a square $K * K$ matrix, $A = a_{ij}$ (state transition probability matrix) where $a_{ij} \geq 0$. Transition probability has been calculated using the following formula

$$P[q_i|q_j] = \frac{N(q_i|q_j)}{\sum_{l=1}^{K} N(q_l|q_j)} \tag{7}$$

Where $N(q_i, q_j)$ is number of times a turn of class q_i follows a turn page of class q_j.

The initial state distribution ($\prod = \pi_i$) should also be defined as

$$\pi_i = P[q_1 = q_i] \qquad where \quad 1 \leq i \leq K \wedge \pi_i \geq 0 \tag{8}$$

In emission probability the observation x_t depends only on the present state q_t

$$P[x_t = x_j | q_t = q_i] = b_{ij} \tag{9}$$

This defines a $K * N$ matrix (B), which we call as emission probability matrix, $B = b_{ij}$ where $bij \geq 0$. For emission probabilities, there can be a number of possible formulations. Looking at the sentence feature vector, we take the view that the probability of a sentence vector being generated by a particular cluster is the product of the probabilities of the index terms in the sentence occurring in that cluster according to some distribution, and that these term distribution probabilities are independent of each other. Hence emission probability is b_{ij} is calculated as

$$P[x_j|q_i] = \prod_{t=1}^{|x_j|} P[\omega_j^t|q_i] \tag{10}$$

Where $|x_j|$ is the number of words in x_j and ω_j^t is the t^{th} term in x_j. Probabilty of a word given a state is defined as

$$P[\omega^l|q_i] = \frac{1 + N(\omega^l, q_i)}{|V| + \sum_{m=1}^{|V|} N(\omega^m, q_i)} \tag{11}$$

Where, $N(\omega^l, q_i)$ is the number of occurrences of word ω^l in pages of class q_i. $|V|$ is the vocabulary size. $1/|V|$ is Dirichlet prior over the parameters and plays a regularization role for the rare words.

3.2.4 Procedure Generation

We learn the HMM until it converges. Then the HMM is used to generate most frequent possible sequences of turns. We have generated procedures of length of L, where L is the average length of the calls belongs to specific topic. Finding the procedure that gives highest total probability is a dynamic programming problem, which is very inefficient. A greedy approach is taken instead, which gives approximate result. But as many procedures have been considered and then highest probability procedures have been picked this approximation gives good result. We have used the end probability along with transition and initial probability. End probability is the probability of states that a call ends with.

3.2.5 Recommending Procedure to Agent

Given a partial call transcript, we convert it into a sequence of SPTS clusters, and use it to find relevant procedures which would then be displayed to the agent. Extracted procedures may be presented to the agent as sequences of sets of keywords which best describe each step in the procedure. When a call comes to the agent, at time instance t a procedure is prompted to the agent if the *Relevance* function on that partial call and that procedure returns true. Relevance function is defined as

$$\begin{aligned} Relevance(P_j, C_i^t) &= true, \qquad \text{if } isSubSequence(P_j^{\overline{\frac{m_j * n_t}{l}}}, C_i^t) = true \\ &= false, \qquad otherwise \end{aligned} \tag{12}$$

Where l is average length of call, n_t is the length of partial call, m_j is the length of procedure, $isSubSequence()$ checks if first argument is contained in the second argument.

3.2.6 Evaluation of Recommendation

Let a call C_i, upon completion, be represented by the sequence of SPTS clusters $(S_1, S_2, \cdots, S_n)$ where n is the length of the call. Let $P = P_1, P_2, \cdots, P_w$ be the set of procedures extracted from a historical call corpus using the methods outlined in section above. We define P_C, the set of procedures from P which are employed in the Call C_i as

$$P_{C_i} = P_j|(P_j \in P) \wedge isSubSequence(P_j, C_i) = true \tag{13}$$

At a given time t, using the partial Call C_i^t, a set of procedures $P_{C_i^t}$ can be extracted from the procedure collection using the *Relevance* function. For a completed call C_i, we evaluate the relevance of the procedures retrieved at different points of time (before completion) in the call $(t_1, t_2, \cdots, t_p)$ by measuring the correspondence between each of the sets $(P_{C_i^{t_1}}, P_{C_i^{t_2}}, \cdots, P_{C_i^{t_p}})$ and the known set of relevant procedures for the completed call,P_{C_i}. We use the classical measures of Precision, Recall and F-Measure to evaluate this correspondence. $PREC^P_{C_i^t}, REC^P_{C_i^t}$ *and* $F^P_{C_i^t}$, the Precision, Recall and F-Measure for the Partial Call C_i^t , using the procedure collection P are calculated as below:

$$PREC^P_{C_i^t} = \frac{|P_{C_i^t} \cap P_C|}{|P_{C_i^t}|} \tag{14}$$

$$REC^P_{C_i^t} = \frac{|P_{C_i^t} \cap P_C|}{|P_C|} \tag{15}$$

$$F^P_{C_i^t} = \frac{2 * PREC^P_{C_i^t} * REC^P_{C_i^t}}{PREC^P_{C_i^t} + REC^P_{C_i^t}} \tag{16}$$

3.3 Experimental Results

We used the call transcripts obtained using an ASR (Automatic Speech Recognition) system from the internal IT helpdesk of a company. The calls are about queries regarding various issues like Lotus Notes, Net Client etc. The prefixes of sentences in the transcripts are either "Customer" or "Agent", depicting the role of the speaker. Calls are one-to-one conversations between an agent and a customer. The data set has about 4000 calls containing around 68000 sentences. The ASR system used for generating transcripts has an average Word Error Rate of 25% for Agent sentences and 55% for customer sentences. We have done the topical clustering taking K=50. For each topical clusters HMM is learned separately. Table 3 shows the F-Measure result of three topical cluster. In the table $x\%$ (where $x = 20, 40, 60, 80$) means considering the call when it is $x\%$ complete.

Table 3: F-Measure values

Topic	20%	40%	60%	80%
Lotus Notes	0.49	0.69	0.73	0.76
Password	0.45	0.60	0.70	0.81
Serial Number	0.51	0.61	0.73	0.75

4 CONCLUSION

In movie recommendation a hybridization of content based and collaborative filtering based recommendation is proposed. The weights of different attributes of an item are computed from the collaborative social network using regression analysis. Further studies to use this framework on other applications like web social network can give us more confidence on this concept. In agent prompting of call center we propose a method to extract procedural information from contact center transcripts. We define SPTS clusters and taken HMM based probabilistic approach to obtain better SPTS clusters. Procedures have been generated from HMM. We show that these procedures are useful in guiding an agent to the relevant procedure in an agent prompting application. This HMM based approach can be used to segmentation and classification of the call.

References

[1] Internet Movie Database. http://www.imdb.com.

[2] Bruke, R. *Hybrid recommender systems: survey and experiments*, User Modeling and User Adapted Interaction 12 (2002) 331-370.

[3] P. Melville, R.J. Mooney, R. Nagarajan. *Content-Boosted Collaborative Filtering for Improved Recommendations*, Proceedings of the 18th National Conference on Aritificial Intelligence (AAAI-2002), July 2002, Edmonton, Canada.

[4] Deepak P., K. Kummamuru. *Mining Conversational Text for Procedures with Applications in Contact Centers*, to appear in the International Journal on Document Analysis and Recognition (IJDAR) (Special Issue on Noisy Text Analytics), Springer.

[5] S. Roy, L. V. Subramaniam. *Automatic Generation of Domain Models for Call Centers from Noisy Transcriptions*, In ACL-2006.

[6] Pascale Fung, Grace Ngai, and Percy Cheung. *Combining optimal clustering and hidden Markov models for extractive summarization*, in Proceedings of ACL Workshop on Multilingual Summarization, 2003, pp. 29-36.

[7] Frasconi, P., Soda, G., and Vullo, A. *Hidden Markov Models for Text Categorization in Multi-Page Documents*, Journal of Intelligent Information Systems, 18(2/3):195-217. Special Issue on Automated Text Categorization.