Machine Learning Based Recommendation System

Thesis 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
Dr. Niloy Ganguly

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

Dedicated

To My Loving *Parents*

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Kharagpur, India 721302.

CERTIFICATE

This is to certify that the thesis entitled **"Machine Learning Based Recommendation System"**, submitted by **Souvik Debnath, Roll No. 06CS6036** in partial fulfillment for the award of the degree of ***Master of Technology in Computer Science and Engineering***, is a record of bona-fide work done by him, under our supervision, during the period of 2007-2008.

This thesis, our opinion, is worthy of consideration for the award of degree of Master of Technology in accordance with the regulations of this institute.

Dr. Pabitra Mitra
Department Of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
India

Dr. Niloy Ganguly
Department Of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
India

ACKNOWLEDGEMENTS

I extend my ineffable and deepest gratitude to my supervisor **Dr. Niloy Ganguly** and **Dr. Pabitra Mitra** for their esteem supervision, incessant inspiration, prolific encouragement, creative criticisms and also supplying the needs of runaway geniuses throughout the course of this thesis work. Their innovative ideas and invaluable suggestions have introduced me to a very interesting and dynamic field. They have been very much responsible for ensuring that my work progressed smoothly and surely in the right direction. I am grateful to them for the completion of my thesis work.

I am deeply indebted to **Dr. Krishna Kummamuru**, IBM India Research Lab, who guided me with his esteem knowledge at the time of internship which was a part of my thesis work.

This research was supported in part by my family and friends with their constant moral support, love and encouragement throughout my endeavors.

Souvik Debnath
Roll. No. 06CS6036
Department Of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
India

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.

Contents

Chapter 1

INTRODUCTION

Recommendation systems produce a ranked list of items on which a user might be interested, in the context of his current choice of an item. It helps the user to address the problem of information overload and finds the relevant objects for user. Recommendation system has been built for movies, book, communities, news, articles etc. There are two main approaches to build a recommendation system - collaborative filtering and content based [5]. Collaborative filtering recommenders use the opinions of other users to predict the value of items for each user in the community. Whereas content based recommendation systems recommend on basis of the content similarity between objects. There are several advantages and disadvantages of both which have been described in detail at Chapter 2. A hybrid recommender system also can be built by combining collaborative filtering and content based system.

Content based recommendation systems are very much useful when there is no or very less user data available. In that case depending on the similarity between the items the system recommends. The similarity of the content of the items is measured. To find this similarity various machine learning techniques (supervised or unsupervised) can be applied. The content also can be structured or unstructured. The later should be converted to structured content to make the similarity measure easier. Generally the structure of the content of two objects should be same to find the similarity. But in some situation it may possible that out of two objects, partial content is available for the one.

We have built recommender system for two application- Movie and Conversation Text. In movie recommendation 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. We used the structure content (feature set of

director, writer, cast etc.) of movies. To find the similarities between two movies their feature sets are compared. But if we consider cosine similarity like measure that will give same weights to all the features. Though some features can be of more importance and some are of less. All the features may have different weights. These weights have been calculated in supervised technique, where the number of common reviewer has been used as a supervised data to the system. The same carries the collaborative information between two movies. Chapter 3 has the detail of the algorithm and experiments of this weight learning technique.

The second application we have considered conversational text data of 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. This call data are text data which are unstructured. This unstructured data have been converted to structured data as term vector. We have used unsupervised machine learning technique to build this procedure recommendation. Chapter 4 has the detail of the algorithm and experiments of this agent prompting technique.

Chapter 2

RECOMMENDATION SYSTEMS

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)*[5].

1. Collaborative Filtering: 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.

2. Content Based: 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. Content of an item can be structured or unstructured. For structured content extracting features is straightforward. If we consider the content of a movie as director, writer, cast etc., then each of these attribute can be considered as a feature. But in case of unstructured item like text data, deciding on the feature is difficult.

General approach to that is building a term vector of text and considers the terms as features.

Collaborative Filtering can be categorized into three broad types.

1. Active CF: In Active CF, user actively participate to build the recommendation system by giving ratings or direct recommendation opinion on an item. Along with item profile data, user profile data also required to be maintained in Active CF.

2. Inactive CF: Inactive CF captures and analyses user activities like buying sequence or web access nature etc. Inactive CF does not use the user profile data.

3. Item Based CF: In item based CF a team of experts analyses the data of similar items and give direct recommendation. This type of recommendation can be found in various computer magazines.

In the other way, recommendation system can be categorized into two types- *item to item* and *user to user*. Item to item recommendation is mainly based on item data by comparing the item data, but user to user recommendation is mainly based on user data. Active CF is an example of user to user recommendation system. Content based recommendation is an example of item to item recommendation system. Some inactive CF also falls under the item to item recommendation category. If we apply association rule on the purchase history of the items and uses associated item as recommendation, it will be a type of inactive CF and also item to item recommendation. There is no importance of notion of sequence of items in a purchase session. But in some cases, the sequence of the item in a session is also important. For example if we consider the web page hit by a user in a single session, the sequence can be considered to make the recommendation.

It is possible to make the recommendation systems on a collaborative filtering approach or on a content based approach. But there are some advantages and disadvantages of both. Collaborative filtering has these set of well known disadvantages.

1. Cold Start:There needs to be enough other users already in the system to find a match.

2. Sparsity:Most users do not rate most items and hence the user-item matrix is typically very sparse. If there are many items to be recommended, even if there are many users, because the user-ratings matrix is sparse, and it is hard to find users that have rated the same items.

3. First Rater:It is not possible to recommend an item that has not been previously rated. This problem comes for new items mostly. An obscure item also may face this problem.

4. Popularity Bias:CF cannot recommend items to someone with unique tastes. In that case there is a tendency to recommend the popular.

Content based recommendation solves the above problems, but has its own constrains like-

1. CB is possible only for similar items like recommendation of movie. But it is not possible for recommend items on a super store. So this constraint is not a problem for movie recommendation.

2. It requires content that can be encoded as meaningful features. This is also not a constraint for movie recommendation as it has a nice set of features like director, writer, release date, genre etc. which can be easily encoded.

There are various attempts to combine content based system and collaborative filtering depending on the domain, data availability and requirement. A hybrid recommendation system is the combination of content based system and collaborative filtering.

Chapter 3

MOVIE RECOMMENDATION USING SOCIAL NETWORK ANALYSIS

Many websites like IMDB [1] have the movie recommendation feature. Movie recommendation (or any recommendation) can be modeled in two ways. In first model, 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. This kind of recommendation systems can be built using the content data of the items, no user data is required.

$$f(movie) \rightarrow \{movies\} \tag{3.1}$$

In second model, user enters the system and gets the recommendation without hitting a movie. This recommendation can be provided depending on the profile data of the user or the previous web access history of the user. This is an example of personalization where recommendation varies depending on the user's profile. This can be done in both content based and collaborative filtering. In case of content based, a user profile is built without relating that with other user profiles. Depending on the user's previous movie access history or opinion given for movies, a movie profile like user profile can be built, which can be compared for similarity with movies for recommendation. In case of collaborative filtering for a given user, it finds other similar users whose ratings on movies strongly correlate with the current user. It recommends movies which are rated highly by these similar users.

$$f(movies, user) \rightarrow \{movies\} \tag{3.2}$$

There can be three types of data, which can be used to build the movie recommendation -

1. Item factual data: Movie features such as director, actor, writer, genre comes under this category.

2. User demographic data: Profile data of users.

3. Transactional data: This is the data about the movie-user relation. For example the ratings given by the user, or the access history of the user.

3.1 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. Other disadvantages have been described in Chapter 2. Content based system on the other hand does not use any preference data and provides recommendation directly based on similarity of items. Similarity measures computed based on item attributes, are however difficult to define. Human judge similarity between two items based on some latent attributes. The similarity measure should be based on such latent attributes for good quality content based recommendation. Note that the content based recommendation does not suffer from the problem of preference scarcity. 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.

3.2 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

$$S(O_i, O_j) = \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \cdots + \omega_n f(A_{ni}, A_{nj}) \tag{3.3}$$

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 most of the 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 3.1. 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.

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.

Table 3.1: Features Used in Movie Recommendation

Feature	***Type***	***Domain***	***Distance Measure***
Release	Year	YYYY	$\frac{(300-\|Y_1-Y_2\|)}{300}$
Type	String	Movie,TV,VG,V,mini	$T_1 = T_2?1:0$
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)*	Action, Drama etc.	$\frac{\|G_1 \cap G_2\|}{G_{max}}$
Keyword	(String)*	College, Thief etc.	$\frac{\|K_1 \cap K_2\|}{K_{max}}$
Cast	(String)*	(<Name>)*	$\frac{\|C_1 \cap C_2\|}{C_{max}}$
Country	(String)*	India, France etc.	$\frac{\|C_1 \cap C_2\|}{C_{max}}$
Language	(String)*	English,Russian etc.	$\frac{\|L_1 \cap L_2\|}{L_{max}}$
Color	String	Color, B/W	$C_1 = C_2?1:0$
Company	String	<Name>	$C_1 = C_2?1:0$

$E(O_i, O_j)$, 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. $\forall i, \forall j = 1..l \wedge i \neq j$,

$$\omega_0 + \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \cdots + \omega_n f(A_{ni}, A_{nj}) = E(O_i, O_j) \qquad (3.4)$$

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.

3.3 Experimental Results

3.3.1 Dataset Used

The movie database used in our recommendation system consists of 3×10^5 random movies downloaded from the IMDB. A crawler has been written to download the feature data of movies and another to download the list or reviewer for the movies. The downloaded data have been formatted such that they can be processed efficiently. Figure 3.1 gives a system overview of movie recommendation system. 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 3.1).

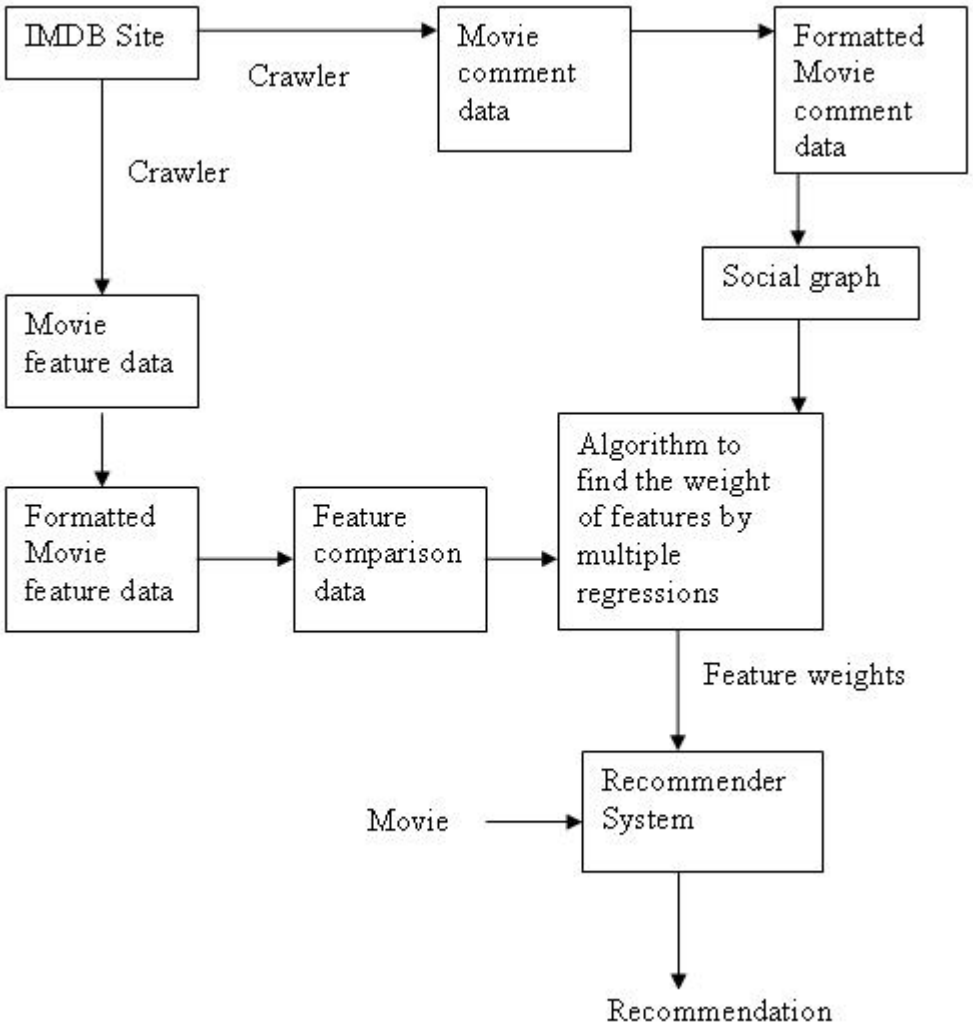

Figure 3.1: System Overview of Movie Recommendation

3.3.2 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 3.2) 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 3.2: 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

3.3.3 Sensitivity of the Feature Weights

We compare our recommendation with the recommendation using the same features but with slightly changed (increasing half of the feature weights by 2% and decreasing half of the feature weights by 2%) weights. We ranked both the recommendation result on basis of their similarity score. With these two different ranking we measure

the *Spearman's rank correlation coefficient*. This experiment we have done with 15 different movies. We found the rank correlation coefficient in the range of 0.96 to 1.0. So we can see the feature weights we have got are not over sensitive to slight changes.

3.3.4 Performance of the Recommender System

We used a set of $3 * 10^5$ Movies to construct the regression equations. The weights obtained are used to recommend on a different set of 10^5 movies. The recommendations are ranked according to the value of similarity measure S (Eq. 3.3). The performance of proposed algorithm is measured using two procedures-

1. Classical *Recall* measure
2. Human Judgement

The Recall measure is defined as

$$Recall = \frac{|\{relevant\ movies\} \cap \{retrieved\ movies\}|}{|\{relevant\ movies\}|} \tag{3.5}$$

For Recall measure the proposed algorithm is compared with pure content based method (considering equal weights for all features), 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.

We have used human judgment [4] to measure the quality of our recommendation. In this method proposed algorithm has been compared with both with pure content based method (considering equal weights for all features) and IMDB recommendation. We have presented 9 movies and 50 recommendations for each movie to ten humans of different ages. 50 recommendations were compiled using 20 recommendations by our method, 20 recommendations by pure content based method, and 10 recommendations by IMDB (downloaded from IMDB site). They have been asked to rate each recommended movie in the scale of 0 to 5 on basis of its similarity to its seed and to exclude the movie if they have not seen that. We got the response from three volunteers only. The result shows that in average over all the volunteers and all the movies, out method scores 2.28 which is slightly better than the pure content based, which scores 2.24. IMDB recommendation scores 2.37, which is because IMDB uses direct users input for its recommendation.

Chapter 4

PROCEDURE RECOMMENDATION TO CALL CENTER AGENT

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. Call centers are the direct point of contact for the customer to the company. It is very important part of service, which is provided by a company to the customer. Because of that most of the company has its call center which may be outsourced to some other company specialized in that work. Hundreds of calls come to each agent of a call center which varies in type and complexity. The interaction between the caller and responder can be considered as the conversation between those two. Conversation text is derived from this conversation. Advances in speech recognition systems and their widespread deployment in call centers for monitoring of agents, large volumes of dialog transcripts are available which are having lots of information inside them.

There are some basic differences between conversation text and usual text. There is always two or more role involved in the conversation text. The sentences of conversation text are usually very short compare to usual text sentences. Sentences of conversation text are very noisy due to use of abbreviations, grammatical error, spelling error (esp. errors introduced by ASR)

A call center handles calls of different nature. Broadly, calls can be classified in three type-

1. Support-related:

- Mobiles: Call dropped, wrong billing - Computers: Not able to access the network, slow response, outlook is not opening, etc.

2. Transactional-based:

- Airline ticket booking, Car rental reservation, Change in mobile monthly rental plan, etc.

3. Exploratory-based:

- Timings of flights from Kolkatta to Delhi, how to optimize mobile monthly bill, etc.

Figure 4.1 shows an example of a conversation of customer and agent in a call center.

```
AGENT (greeting):  Welcome to CarCompanyA. My name is Albert.  How may I help you?
.........
AGENT (details):  Allright may i know the location you want to pick the car from.
CUSTOMER (details):  Aah ok I need it from SFO.
AGENT (details):  For what date and time.
.........
AGENT (rates):  Wonderful so let me see ok mam so we have a 12 or 15 passenger van
avilable on this location on those dates and for that your estimated total for those
three dates just 300.58$ this is with Taxes with surcharges and with free unlimited free
milleage.
.........
AGENT (verify):  alright mam let me recap the dates you want to pick it up from SFO on
3rd August and drop it off on august 6th in LA alright
CUSTOMER (objection_handling):  oh and one more question Is it just in states or could
you travel out of states
.........
AGENT (conclusion):  The confirmation number for your booking is 221 384.
CUSTOMER (conclusion):  ok ok Thank you
AGENT (conclusion):  Thank you for calling CarCompanyA and you have a great day good
bye
```

Figure 4.1: Example Conversation Text

A typical call in a customer service center follows some patterns which consist of various steps such as *"greeting"*, *"getting details from the customer"*, *"issue resolving steps"*, *"verification"*, and *"sign-off"*. Within these broad steps, there are

some domain specific or very particular subject specific information exchanges. For instance, *"checking whether the wireless connection is working"* would be a sub-step which characterizes the topic of the call, which in this case is a wireless connection problem. *"Right-clicking My Computer"* and *"selecting properties"*, on the other hand, is an example of a sub-step which conveys information about the procedure used to resolve an issue. We call such information exchanges that stand for sub-steps in a call as ***sub-procedure text segments (SPTS)***. Call may contain lots of data like *yah*, *I see*, *right*, etc which do not carry any information. A call can be seen as a sequence of SPTS. In that consideration a call should not be represented as a point in the n-dimensional vector space model, where n is the length of term vector. Rather a call becomes a line, which is sequence of some points in vector space and each point is a ***Turn*** of that call. A call is a conversation between customer and agent in which they speaks alternately. In each turn to the speaker whatever sentences the speaker says is defined as a ***Turn***. Figure 4.2 shows the representation of a call as a line in vector space. There are some points in the vector space which are very dense. In other words, many calls passes through those points. Those points actually represent SPTSs. We try to find those points. The points in vector space represent some Turns which can be modeled as bag-of-word. So clustering those points with a bag-of-word approach will give cluster centroids, which can be considered as SPTSs. But, this approach does not consider the sequence information of call. In a call a turn can be thought as an effect of previous turn. To get the effect of sequence into the clustering we have used HMM.

4.1 Related Work and Motivation

We have seen that the call generally follows some patterns. In this work we have tried to extract those patterns and define some procedure from those which can be recommended to an agent when she gets a call online. When an agent gets 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 and the content of call the possible solution should be prompted to the agent automatically, so that the agent can respond to the call efficiently.

There were some previous attempts on call center dialogs mining [6], [7]. 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

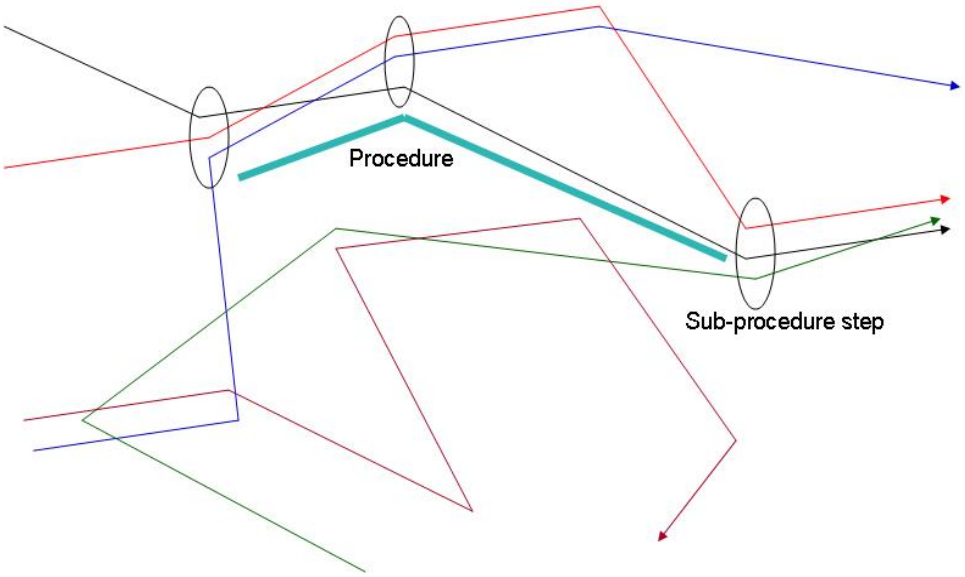

Figure 4.2: Line Representation of Call in Vector Space

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 agent prompting.

4.2 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 [10] are used. The SPTS label sequence, which are found after KMA, are used to learn the HMM. In HMM, SPTS 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. Figure 4.3 shows the flowchart of our algorithm. The details of each steps are described below.

4.2.1 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.

4.2.2 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,

$$G_i = \bigcup_{\forall c \in T_i, \forall l, Speaker(v_l(c)) = "Caller"} v_l(c) \tag{4.1}$$

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

$$H_i = \bigcup_{\forall c \in T_i, \forall l, Speaker(v_l(c)) = "Responder"} v_l(c) \tag{4.2}$$

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*

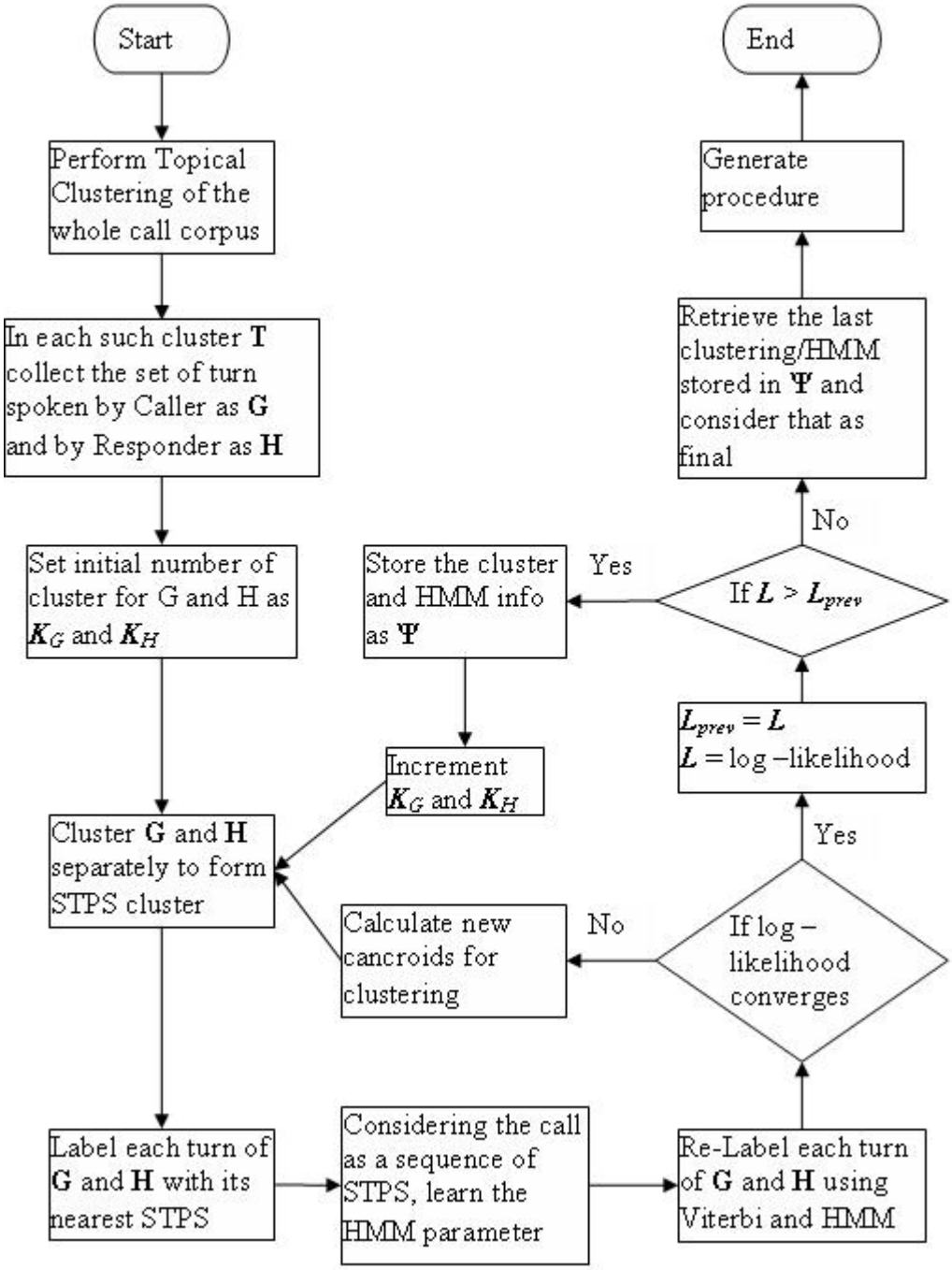

Figure 4.3: Overview of the Algorihm

(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 (Figure 4.4) Q_t is the turn at time instance t and Q_{t-1} is the previous turn and so on.

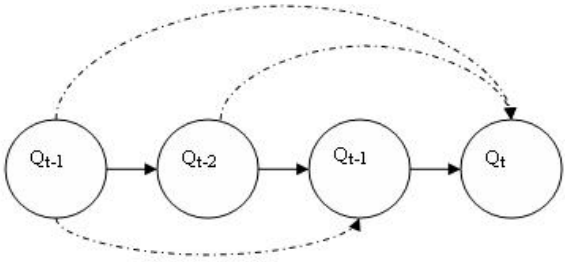

Figure 4.4: Turn Dependence

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.

$$P[Q_t = Q_j | Q_{t-1} = Q_i, Q_{t-2} = Q_l,] = P[Q_t = Q_j | Q_{t-1} = Q_i] \tag{4.3}$$

If we consider the text of the turn as observation, our problem is to find the cluster (SPTS label) of that turn. Hence we can fit a Hidden Markov Model here where the hidden state is the SPTS 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 SPTS clustering together with in an iterative fashion. Here is the sketch (Figure 4.5) 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 clustering of H and G separately
2. Label each turn with nearest cluster
3. Learn HMM Param, $\lambda = (\pi, A, B)$
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

}

Figure 4.5: Sketch of KMA/HMM algorithm

4.2.2.1 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 (λ) requires its three probability measure to be defined, transition probability (A), emission probability (B) and the initial probability (π). $\lambda = (\pi, A, B)$.

In our case initial SPTS 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.

$$P[q_t = q_j | q_{t-1} = q_i] = a_{ij} \qquad where \quad 1 \leq i, j \leq K \tag{4.4}$$

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

$$P[q_i|q_j] = \frac{N(q_i|q_j)}{\sum_{l=1}^{K} N(q_l|q_j)} \tag{4.5}$$

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* (π) should also be defined as

$$\pi_i = P[q_1 = q_i] \qquad where \quad 1 \leq i \leq K \wedge \pi_i \geq 0 \tag{4.6}$$

In *Emission Probability* the observation x_t depends only on the present state q_t

$$P[x_t = x_j | q_t = q_i] = b_i(x_j) = b_{ij} \tag{4.7}$$

This defines a $K * N$ matrix (B), which we call as emission probability matrix, $B = b_{ij}$ where $b_{ij} \geq 0$. 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

$$P[x_j|q_i] = \prod_{t=1}^{|x_j|} P[\omega_j^t|q_i] \tag{4.8}$$

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

$$P[\omega^l|q_i] = \frac{1 + N(\omega^l, q_i)}{|V| + \sum_{m=1}^{|V|} N(\omega^m, q_i)} \tag{4.9}$$

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.

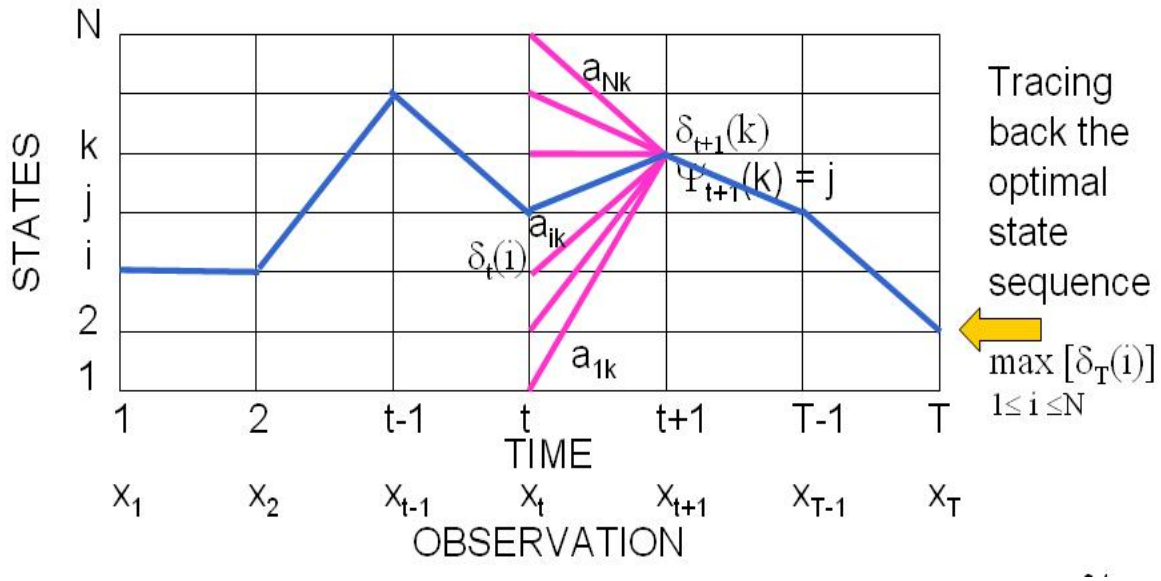

Figure 4.6: Viterbi

4.2.2.2 Viterbi Algorithm

After we learn the HMM parameter the steps we need to do is to re-label all the turns of each sequence with its appropriate cluster label. Hence we have to find a hidden state sequence $\Gamma^* = q_1, q_2, \cdots, q_T$ such that probability of occurrence of the observation sequence $X_s = x_1, x_2, \cdots, x_T$ from the state sequence is greater than that from any other state sequence. In other words, the problem is to choose the most likely path path $(q_1, q_2, \cdots, q_T)$ that maximizes the probability $\mathrm{P}(\Gamma^*|X_s, \lambda)$. The most likely state sequence can be obtained by running Viterbi's algorithm[10]. Viterbi algorithm is a Dynamic Programming algorithm.

We define $\delta_t(i)$ as the highest probability path that ends in state q_i at t^{th} observation.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = q_i, x_1, x_2, \ldots, x_t|\lambda) \tag{4.10}$$

By induction we can have,

$$\delta_{t+1}(k) = [\max_{1 \le i \le N} [\delta_t(i) * a_{ik}]] * b_k(x_{t+1}), \quad \text{with} \quad 1 \le k \le N \tag{4.11}$$

Where a and b is same as it is defined in eq.4.4 and eq.4.7. Also we define ψ as

$$\psi_{t+1}(k) = \arg\max(\delta_t(i) * a_{ik}) \tag{4.12}$$

Figure 4.6 gives an idea how viterbi progresses.

Here is a sketch of the Viterbi Algorithm.

1. Initialization

For $1 \leq i \leq N$

$\delta_1(i) = \pi_i * b_i(x_1)$

$\psi_1(i) = 0$

2. Recursive computation

For $2 \leq t \leq T$

For $1 \leq j \leq N$

$\delta_t(j) = \max_{1 \leq i \leq N}[\delta_{t-1}(i) * a_{ij}] * b_j(x_t)$; // Probability of t^{th} observation is of state j

$\psi_t(j) = \arg\max_{1 \leq i \leq N}(\delta_{t-1}(i) * a_{ij})$; // State of t^{th} observation if $t+1^{th}$ observation is j

3. Termination

$P^* = \max_{1 \leq i \leq N}[\delta_t(i)]$;

$q_T^* = \arg\max_{1 \leq i \leq N}[\delta_T(i)]$;

4. Backtracking

For $t = T-1$ down to 1

$q_t^* = \psi_t(q_{t+1}^*)$;

Hence P^* gives the required state-optimized probability, and $\Gamma^* = (q_1^*, q_2^*, \ldots, q_T^*)$ is the optimal state sequence.

4.2.2.3 Expectation Maximization for Convergence

In the algorithm sketched in 4.5, we do the KMA clustering and learn the HMM parameter together. After KMA clustering we learn the HMM model and using HMM, we again do the clustering of the turn using the Viterbi and HMM. Depending on some convergence condition we stop the iteration else continue it with the new cluster centroids for KMA clustering. These new cluster centroids are calculated from the clusters we get after re-labeling the turns with Viterbi. This algorithm can be seen as an Expectation- Maximization algorithm. The two steps in one iteration of the EM algorithm are as follows:

E-step: re-estimates the expectations based on the previous iteration In 4.5 steps 4 is E-steps of the iteration. In this step we are re-estimating the cluster label for each turn using Viterbi.

M-step: updates the model parameters to maximize the log-likelihood In 4.5 steps 1,2,3,5 are M-steps of the iteration. In these steps we are measuring the parameters of our model.

Our HMM model (λ) consist of three basic parameters (π, A, B) and the two number as,

N - Total number of turns. (Specific to one topical cluster)

K - Total number of clusters.

The data corpus contains all the turns specific to one topical cluster.

$$X = \{x_1, x_2, \ldots, x_N\} \tag{4.13}$$

Viterbi gives the highest probability class for each document with its probability $P(x, q|\lambda)$. Log-likelihood that the data corpus is generated from the model λ can be defined as-

$$log(L(X|\lambda)) = log \prod_{i=1}^{N} P(x, q|\lambda) \tag{4.14}$$

We stop the iteration when Log-likelihood ($log(L(X|\lambda))$) converges.

4.2.3 Procedure Generation

We learn the HMM until Log-likelihood function 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. We have generated many procedures using greedy and then all the procedures have been ranked on the generation probability values. Then highest probability procedures have been considered. So this approximation gives good result. For generation of procedure along with initial and transition probability, end probability also has been considered. End probability is the probability of states that a call ends with. The end state distribution (ξ) should also be defined as

$$\xi_i = P[q_T = q_i] \qquad where \quad 1 \leq i \leq K \quad \wedge \quad \xi_i \geq 0 \tag{4.15}$$

Where T is the lenght of the sequence.

4.2.4 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

$$\begin{aligned} Relevance(P_j, C_i^t) &= true, \qquad \text{if } isSubSequence(P_j^{\frac{m_j * n_t}{l}}, C_i^t) = true \\ &= false, \qquad otherwise \end{aligned} \tag{4.16}$$

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.

4.2.5 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 4.2.3. We define P_C, the set of procedures from P which are employed in the Call C_i as

$$P_{C_i} = P_j | (P_j \in P) \wedge isSubSequence(P_j, C_i) = true \tag{4.17}$$

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. $PREC_{C_i^t}^P, REC_{C_i^t}^P \quad and \quad F_{C_i^t}^P$, the Precision, Recall and F-Measure for the Partial Call C_i^t , using the procedure collection P are calculated as below:

$$PREC_{C_i^t}^P = \frac{|P_{C_i^t} \cap P_C|}{|P_{C_i^t}|} \tag{4.18}$$

$$REC_{C_i^t}^P = \frac{|P_{C_i^t} \cap P_C|}{|P_C|} \tag{4.19}$$

$$F^P_{C^t_i} = \frac{2 * PREC^P_{C^t_i} * REC^P_{C^t_i}}{PREC^P_{C^t_i} + REC^P_{C^t_i}} \tag{4.20}$$

4.3 Experimental Results

4.3.1 Dataset Used

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.

4.3.2 Topical Clustering

In a call center a topic may be specific to some domain or so issue with an application, or may be related to an application in itself. The number of such topic was unknown to us. As we had total about 4000 calls, we started with number of topical cluster, $K = 80$, considering each cluster contains 50 calls in average. Each call has been represented as a term vector after stop word removal and stemming. The calls have been clustered by K-Mean Algorithm. After getting the clusters, for each cluster the term distribution has been analyzed. Looking into frequent wordlist for each cluster, it is found many clusters represent same topic. So we reduced the number of clusters and by experiments and trial and error we decided the value of $K = 50$. Using the top 5 frequent word of each cluster the topic of the clusters has been decided. Some domain have been found in more that one cluster that is because diffrent issues of same domain have clustered. There were some frequent words like *password* which were present in many clusters. Some of the clusters could not be identified to any topic because of the lack of representative meaningful keywords.

4.3.3 Comprehensiveness of Procedures

Comprehensiveness can be quantified as the fraction of the length of calls that the procedures span. Our retrieval mechanism assumes that procedures are comprehensive. This assumption is the basis of the *relevant prefix length* calculation in Section 4.2.4. In this subsection, we validate this assumption. The *Span* of a procedure

$P_j = (S^1, S^2, \ldots, S^n)$ that is contained within a call C_i represented as an SPTS Cluster sequence $(S_1, S_2, \ldots, S_m)$ is computed as

$$\begin{aligned} Span(C_i, P_j) &= \frac{max_k\{k|S_k = S^n\} - min_k\{k|S_k = S^1\}}{m}, \text{ if } isSubSequence(P_j, C_i) = true \\ &= 0.0, \qquad otherwise \end{aligned} \tag{4.21}$$

Thus, the span is computed as the fraction of the call C that the procedure P_j covers. The span of a call C_i over a collection of procedures $P = P_1, P_2, \ldots, P_k$ is computed as

$$Span(C_i, P) = \arg\max_j(Span(C_i, P_j)|P_j \in P) \tag{4.22}$$

To compute the span of a call collection (such as a topical cluster) over a set of procedures, we take the cardinality weighted average of the span of each call over the procedure collection (such as the collection of procedures extracted from the topical cluster) across the calls in the collection.

The results of the span analysis are presented in Table 4.1. The results show that procedure collections usually span at least 85% of the call. This validates our assumption that *procedure collections tend to span most of the call* that contain them and thus gives us more confidence procedure generation algorithm.

Table 4.1: . Span Analysis of Procedures

Topical Cluster	***Span of calls in the Topical Cluster***
Lotus Notes	0.8933
Password	0.8504
Serial Number	0.8803

4.3.4 Evaluation of Procedure Recommendation

For each topical cluster, and for each call, we evaluate the Precision, Recall and F-Measure as defined in Section 4.2.5 on completion of 20%, 40%, 60% and 80% of the call. For each topical cluster, we take the cardinality weighted average of these measures across calls in that cluster. Table 4.2 shows the F-Measure result of three topical clusters. In the Table 4.2 x% (where $x = 20, 40, 60, 80$) means considering the call when it is x% complete. As can be seen from the figures, certain topical clusters achieve an F-Measure of around 0.45 even after seeing just 20% of the call. This shows that our method of retrieving procedures provides a very good accuracy. These results show that our procedure retrieval mechanism would work well for guiding the agent quickly to the desired procedure.

Table 4.2: 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

In our turn clustering algorithm we iterate the KMA and HMM unless it satisfies the convergence condition. We compare our F-Measure result with another version of algorithm, which doesn't have iteration and learn the HMM in the first iteration and consider that as final. This experiment is done to check whether our hypothesis of iterative learning is effective or not. From this experiment it is seen that at the 20% completion of call, the F-Measure does not differ much between these two versions of algorithm, but subsequently at time instance of 40%, 60%, 80% our algorithm with iterations out-outperforms the version with single iteration. Thus we can say that iterative learning gives a better and more effective system. Figure 4.7, 4.8, 4.9 show the comparisons for three topical clusters.

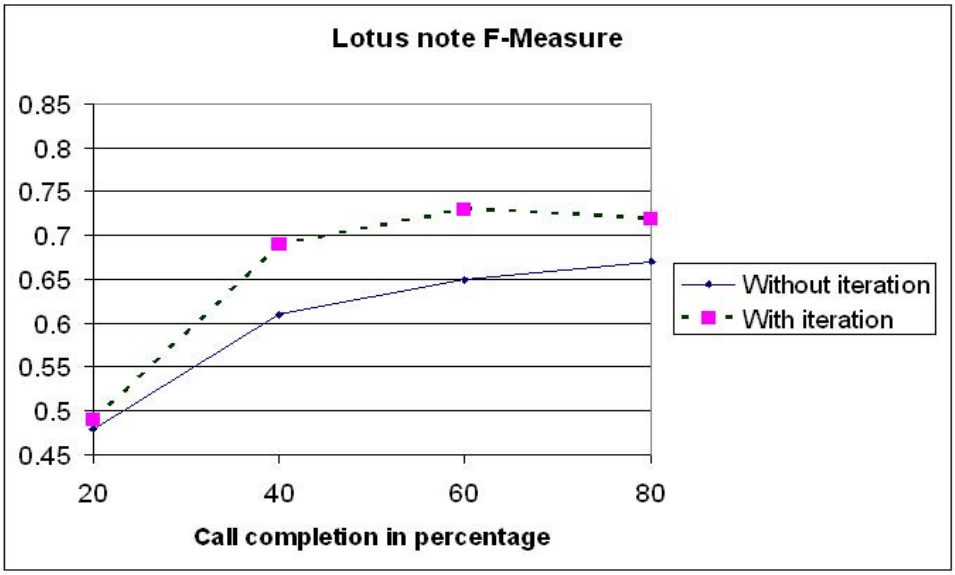

Figure 4.7: F-Masure: Lotus Notes

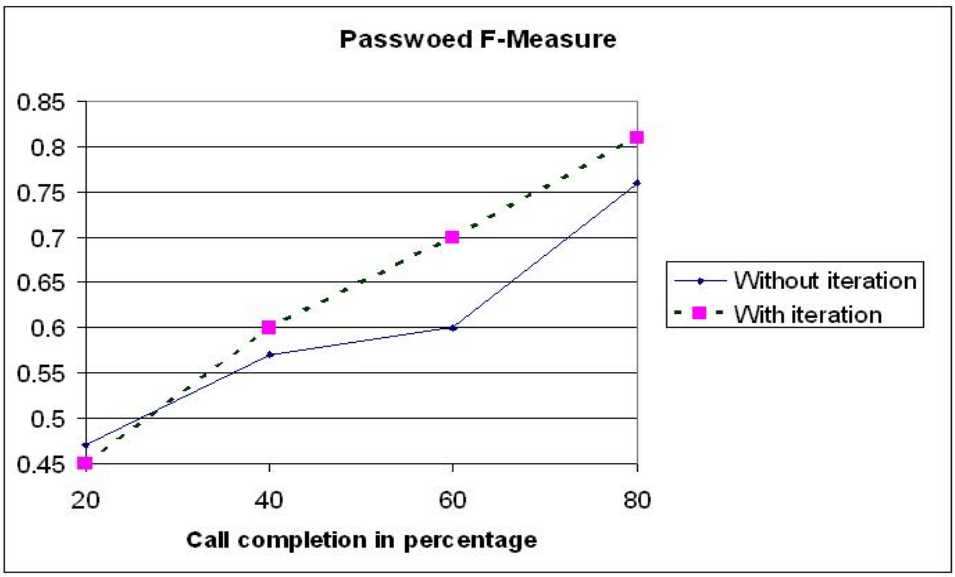

Figure 4.8: F-Masure: Password

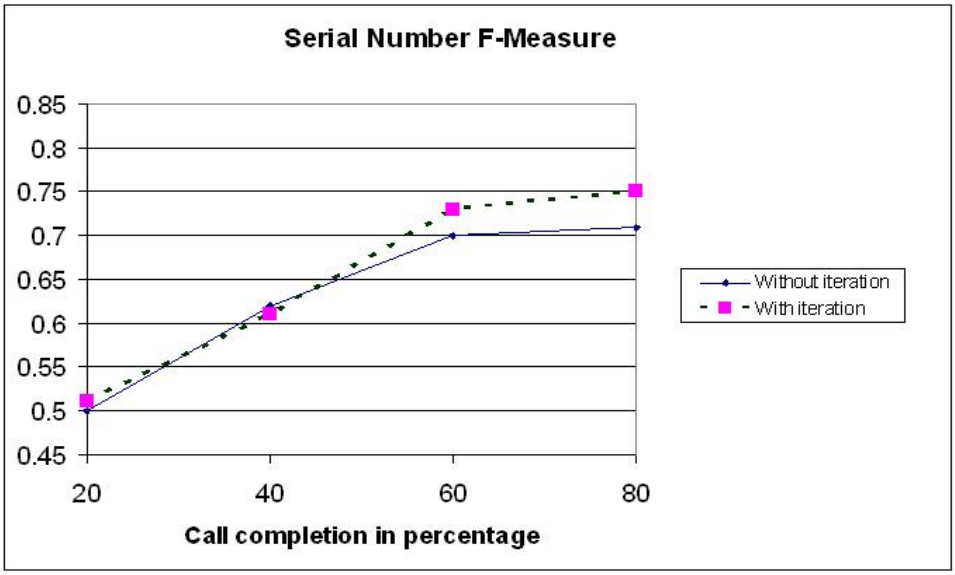

Figure 4.9: F-Masure: Serial Number

Chapter 5

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.

Bibliography

[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] Fleischman, M. and Hovy, E. *Recommendations without User Preferences: A Natural Language Processing Approach*, Intelligent User Interfaces (IUI). Miami Beach, Florida, 2003.

[5] Adomavicius, G., Tuzhilin, A. *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions*, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 734-749.

[6] 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.

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

[8] 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.

[9] 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.

[10] G. D. Forney Jr. *The Viterbi algorithm*, Proceedings of the IEEE 61(3):268278, March 1973.