STANDARDIZATION OF REFERENCES USING HIDDEN MARKOV MODEL

by

Swamynathan Sambamurthy

Bachelor of Technology (IT)
University of Madras, India
2004

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2008
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
The Thesis prepared by

SWAMYNATHAN SAMBAMURTHY

Entitled

STANDARDIZATION OF REFERENCES USING HIDDEN MARKOV MODEL

is approved in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE IN COMPUTER SCIENCE

Examination Committee Chair

Dean of the Graduate College
ABSTRACT

Standardization of References using Hidden Markov Model

by

Swamynathan Sambamurthy

Dr. Kazem Taghva, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

In general, technical papers are augmented with a list of bibliographic citations to support the arguments and the merits of the approach presented. Each and every citation is made up of parts like author, journal, volume, book etc. Extracting the parts of the citation from a written document and properly separating into its parts is the problem that is being addressed in this thesis.

We use an Information Extraction (IE) technique based on Hidden Markov Model (HMM) to solve this problem. This solution consists of the design of an HMM, the training of the HMM with tagged data, and an implementation of Forward Chaining algorithm for extraction of citation parts. Our test on a collection of 150 citations has recall and precision of 0.8 and 0.81 respectively.
# TABLE OF CONTENTS

**ABSTRACT** ............................................................................................................................................ iii

**LIST OF FIGURES** ............................................................................................................................... vi

**ACKNOWLEDGEMENTS** ........................................................................................................................ vii

**CHAPTER 1  INTRODUCTION** ........................................................................................................... 1

1.1 Information Retrieval ........................................................................................................................ 1
1.2 Information Extraction ....................................................................................................................... 1
1.3 Machine Learning .............................................................................................................................. 2
1.4 What is the aim? .................................................................................................................................. 2

**CHAPTER 2  REVIEW OF HIDDEN MARKOV MODELS AND ITS BUILDING BLOCKS** ............... 4

2.1 Probability ......................................................................................................................................... 4

2.1.1 Sample Space ............................................................................................................................... 4
2.1.2 Event ........................................................................................................................................... 4

2.1.2.1 Mutually Exclusive Event ........................................................................................................ 5
2.1.3 Joint Probability ........................................................................................................................... 5
2.1.4 Marginal Probability .................................................................................................................... 5
2.1.5 Conditional Probability .............................................................................................................. 5

2.2 Hidden Markov Models .................................................................................................................. 5

2.2.1 Definition ..................................................................................................................................... 5
2.2.2 What does HMM consists of? ..................................................................................................... 5
2.2.3 Diagrammatic representation of HMM ....................................................................................... 6
2.2.4 How does an HMM work? ......................................................................................................... 6
2.2.5 Markov Property ....................................................................................................................... 9
2.2.6 A working example of an HMM ............................................................................................... 10

**CHAPTER 3  DYNAMIC PROGRAMMING AND ITS ROLE IN HMM** ............................................. 13

3.1 Dynamic Programming (DP) .......................................................................................................... 13
3.2 The three basic problems of HMM and its solution using DP ....................................................... 13

**CHAPTER 4  DESIGN OF HMM FOR THE STANDARDIZATION OF REFERENCES** ..................... 20

4.1 Purpose of the experiment ............................................................................................................... 20
4.2 Technology used ............................................................................................................................... 21
4.3 Deciding the states ........................................................................................................................... 21
4.3.1 State identification .................................................................................................................... 21


<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Diagrammatic Representations of HMM</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>State Transition when current state is S3</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>State Transition from S3 to S2</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Working Example of HMM</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Viterbi Algorithm</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of Viterbi Algorithm</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>Working of Viterbi Algorithm</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Learning curve of HMM</td>
<td>19</td>
</tr>
<tr>
<td>5.1</td>
<td>Training Data</td>
<td>28</td>
</tr>
<tr>
<td>5.4</td>
<td>Output</td>
<td>38</td>
</tr>
<tr>
<td>7.1</td>
<td>Partial HMM design</td>
<td>41</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I would like to dedicate this thesis to my Spiritual GURU, without whom this work would have not been possible. I owe a deep sense of gratitude to Dr. Kazem Taghva, my Advisory Committee Chair, for his thought provoking, painstaking guidance at every stage of my work. His discerning comments and constructive suggestions were very valuable and of great help in completing this thesis. His profound knowledge, expertise and patience to my long hours of dedicated research meetings have been a great source of inspiration and encouragement for me in this endeavor. I am also thankful to Dr. Thomas Nartker, Dr. Yoohwan Kim and Dr. Venkatesan Muthukumar for being a part of my Advisory Committee.

I express my profound sense of gratitude to the Department of Computer Science, and the UNLV Library, for having provided excellent facilities and teaching assistantships, throughout my work.

I sincerely thank Dr. Ajoy. K. Datta for his guidance and help throughout my Masters program.

Further, I would like to thank my Mother, my Fiancée, my sister Mrs. Meenakshi Seshan, my brother-in-law Mr. Seshan and my brothers Mr. Arun Prasath, Mr. Vel Murugan, Mr. Ramakrishnan, Mr. Kirankumar Jayakumar and Mr. V.P.Deepak for motivating me personally in all my endeavors.
I would like to take this opportunity to thank all my friends, especially Mr. Kunal Metkar for his timely encouragements and to Mr. Joe Cameron for his valuable suggestions for my write up.
CHAPTER 1

INTRODUCTION

1.1 Information Retrieval

The science which deals with searching for documents or information that is present inside documents or metadata of the documents is known as Information Retrieval (IR). Apart from that it also means searching a data from database where stand-alone or relational or hyper-textually networked, such as WWW. Some of the most vivid applications of IR are search engines such as Google, Yahoo Search etc.

The process of retrieving information starts when a user queries a system for specific information pertaining to his need. An example of querying is typing a search string in a search engine. The output will not be a single object. It may result displaying diverse objects having different degree of relevancy.

The objects that are stored in the database are matched with the queries given by the user. The data object being an image or text document or video is totally application dependant [1].

1.2 Information Extraction

Information Extraction is a kind of Information Retrieval. The process of finding structured, or unstructured or partially structured data from a document is known as Information Extraction (IE). A visible application of IE is parsing through a
1.3 Machine Learning

Machine learning is devising and designing the algorithms and methodologies that make the system to "learn". Machine learning research primarily targets on extracting necessary information from the data by computational and statistical methods. Hence, they are closely related to data mining, statistics, theory of computation etc [1].

Some of the applications of machine learning are search engines, pattern recognition, speech recognition, classifying DNA sequences, game playing, robot locomotion, etc [1].

Information Extraction can be envisioned in machine learning perspective. Some of the varieties of machine learning techniques used in IE are boosting, rule-learning, sequential classification methods such as Hidden Markov Models, structured support vector machines, conditional random fields.

1.4 What is the aim?

In order to support the merits and arguments of the approach put forth in technical papers, a list of bibliographic citations are listed at the end of the technical paper as "References". Some of the recommended styles to quote these citations are Modern Language Association (MLA) [2], American Psychological Association (APA) [3], or American Mathematical Society’s AMS-LaTex style. Despite these recommended styles in citations, there are citations that come in style that aren’t considered as standards. This occurs because, not all the authors follow the recommended citation style.
There are several formatting tools available to format the citations. One of the most commonly used formatting tools is BibTex [4]. The sources are cited in a consistent manner by making the bibliographic information separate from the presentation information using the BiBTex. In order to enter a record in BibTex, one has to enter each part of the citation separately, Ex: Author, Title, Date, Page Number etc. Extracting the citation parts properly and then separating it into its parts is the main aim of this project. For example, consider the following citation.

Rabiner and Juang, An Introduction to Hidden Markov Models, IEEE, vol. 3, 4-16, 1986

This citation should be parsed and it should extracted the following way

Author = Rabiner, Juang
Title = An Introduction to Hidden Markov Model
Journal = IEEE
Volume = 3
Page = 4 – 16
Date = 1986

Because of the style differences, the citation parts may or may not occur in the same order. Extracting the citation parts manually from the document is a mammoth task, as citations vary from document to document. In this project, an HMM was designed and trained to perform to extract the citations to its parts. The HMM should be trained with training data to make it “learned” enough to extract the information when a test data is given.
CHAPTER 2

REVIEW OF HIDDEN MARKOV MODEL AND ITS BUILDING BLOCKS

The fundamentals of probability play a major role in building a Hidden Markov Model. Thus, reviewing the same may provide a deeper insight towards Markov Models when we talk about it in the later part.

2.1 Probability

The likelihood that something will occur is known as probability. Probability is primarily used in mathematics, statistics etc. [1]

2.1.1 Sample Space

The set containing all possible outcomes is known as sample space.

For example, consider flipping a coin. The possible outcomes are either head (H) or tails (T). Hence the sample space $S = \{H, T\}$. Incase of rolling a die, the sample space would be $S = \{1, 2, 3, 4, 5, 6\}$.

2.1.2 Event

An Event ‘E’ is the subset of the sample space $S$. For example $E = \{1, 3, 5\}$ denotes that the event that an odd number appears on the roll.
2.1.2.1 Mutually Exclusive Events

Consider that there are n events E1, E2, ..., En. If occurrence of one event Eo implies that the rest of the events do not occur, then the event Eo is said to be mutually exclusive with other events. For example, an outcome of the coin cannot be head and tails at the same time. Therefore if A and B are two mutually exclusive events, then AB = ∅ (Null Event).

2.1.3 Joint Probability

The probability of two events in conjunction is known as joint probability. It is written as P(A, B).

2.1.4 Marginal Probability

The probability of an event without considering the other events is known as its marginal probability. Marginal probability of the event A can be written as P(A).

2.1.5 Conditional Probability

The probability of an event A, given another event B is known as conditional probability. It is written as P(A|B).

2.2 Hidden Markov Models (HMM)

2.2.1 Definition:

An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [5].

2.2.2 What does HMM consists of?

An HMM is a probabilistic automaton with Markov property [5].
An HMM consists of

- A set of states, $S = \{s_1, \ldots, s_n\}$.
- A set of observation symbols, $O = \{w_1, w_2, \ldots, w_m\}$.
- Probability of a state emitting the symbol ‘$w$’, $P(w|s)$.
- Probability of moving from $i$th state to $j$th state, $P(S_i, S_j)$.
- Set of start states, to which initial probabilities are assigned.

2.2.3 Diagrammatic representation of HMM

![Diagram of HMM]

2.2.4 How does an HMM work?

1. There are $N$ states in an HMM, from $S_1, S_2, \ldots, S_n$.
2. There are $m$ discrete time stamps from $t = 0$, $t = 1$ to $t = m$.
3. On the $t^{th}$ time stamp, the system is in one of the available states call it $q_t$.
4. Note that, $q_t \in \{S_1, S_2, \ldots, S_n\}$.
The next state is randomly chosen during each time stamp.

S3 is the current state.

\[ N = 3 \]
\[ t = 0 \]
\[ q_t = q_0 = S_3 \]

Figure 2.2

If we make a move from S3 to S2, then the state details changes to what is given in figure 2.3
In HMM, the probability distribution for the next state is determined by the current state. Table 2.1, tabulates the transition probability from one state to another state in the HMM, drawn as Figure 2.1.

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(q_{t+1} = S_1 \mid q_t = S_1)$</td>
<td>0</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_2 \mid q_t = S_1)$</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_3 \mid q_t = S_1)$</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_1 \mid q_t = S_2)$</td>
<td>0.3</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_2 \mid q_t = S_2)$</td>
<td>0.2</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_3 \mid q_t = S_2)$</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_1 \mid q_t = S_3)$</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_2 \mid q_t = S_3)$</td>
<td>0.2</td>
</tr>
<tr>
<td>$P(q_{t+1} = S_3 \mid q_t = S_3)$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table 2.1
2.2.5 Markov Property

The key aspect of HMM is that it doesn’t keep track of history more than one event. That is $P(q_{t+1} = S_j \mid q_t = S_j) = P((q_{t+1} = S_j \mid q_t = S_j), \text{any other earlier conditions})$.

To explain it in detailed, consider the table 2.2

<table>
<thead>
<tr>
<th></th>
<th>$P(q_{t+1} = S_1 \mid q_t = S_1)$</th>
<th>$P(q_{t+1} = S_2 \mid q_t = S_1)$</th>
<th>......</th>
<th>$P(q_{t+1} = S_N \mid q_t = S_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A_{11}$</td>
<td>$A_{12}$</td>
<td></td>
<td>$A_{1N}$</td>
</tr>
<tr>
<td>2</td>
<td>$A_{21}$</td>
<td>$A_{22}$</td>
<td></td>
<td>$A_{2N}$</td>
</tr>
<tr>
<td>3</td>
<td>$A_{31}$</td>
<td>$A_{32}$</td>
<td></td>
<td>$A_{3N}$</td>
</tr>
<tr>
<td>N</td>
<td>$A_{N1}$</td>
<td>$A_{N2}$</td>
<td></td>
<td>$A_{NN}$</td>
</tr>
</tbody>
</table>

Table 2.2

$A_{ij} = P(q_{t+1} = S_j \mid q_t = S_i)$

1

1 is the markov property.
2.2.6 A working example of an HMM [6]

Let us see how an HMM works in detail. Consider the Figure 2.4. The HMM consists of 3 states, and every time state transition occurs from one state to another state after emitting any of the symbols ‘a’, ‘b’, ‘c’. So, an HMM has transition probabilities and symbol probabilities. Now given the symbol sequence a,a,b,c, we have to determine the most probable sequence of states involved in emitting these sequences of symbols.

![Diagram of HMM with transition probabilities]

The state transition probabilities are given in the Figure 2.4. The symbol probabilities are tabulated in Table 2.3, Table 2.4, and Table 2.5.
Now, let us try to calculate the probability of the sequence ‘a’, ‘a’, ‘b’, ‘c’. One key note to be made before doing this is that there are more than one way through which this sequence can be produced. However, the probability of each way differs. The path which gives the maximum probability is taken as the most probable path.
Considering State 1 as start state, there are 3 possible paths through which this symbol sequence can be produced. They are:

(a) 1, 1, 2, 3
(b) 1, 2, 3, 3
(c) 1, 3, 3, 3

The calculations are done below.

1. \( P(a, a, b, c) \) through 1, 1, 2, 3.
\[
0.8 * 0.5 * 0.8 * 0.3 * 0.6 * 0.5 * 0.1 = 0.004068
\]

2. \( P(a, a, b, c) \) through 1, 2, 3, 3
\[
0.8 * 0.3 * 0.2 * 0.5 * 0.3 * 1 * 0.1 = 0.00072
\]

3. \( P(a, a, b, c) \) through 1, 3, 3, 3
\[
0.8 * 0.2 * 0.7 * 1 * 0.2 * 1 * 0.1 = 0.00224
\]

Since calculation (1) has more probability, the most probable path 1, 1, 2, 3. Here, only the symbol sequences are observable and the states are hidden.
CHAPTER 3

DYNAMIC PROGRAMMING AND ITS ROLE IN HMM

3.1 Dynamic Programming (DP)

Dynamic programming is a technique of solving problems which exhibits the properties of overlapping sub-problems. It takes less time than the time expensive naïve methods [1]. Dynamic programming tends to break the major problem into sub-problems and chooses the best solution in the sub-problems beginning from the smaller in size.

The main characteristics of Dynamic Programming algorithms are

1. Application of "principle of the best" strategy, so that in every step, more and more choices are rejected and only the ones that provide the best solution to the sub-problems stays.

2. The solutions for the sub-problems are stored internally for the future use.

3.2 The three basic problems of an HMM and its solutions using DP

Problem 1

Given the observation sequence O, where O = o1, o2, ..., oo, and an HMM Model $\lambda$, where $\lambda = (A, B, \pi)$. How the probability of O is calculated?

This problem is also known as evaluation problem.
Solution using Forward Algorithm

The probability of an observation sequence is the sum of probabilities of all possible states sequence in the HMM. Thus, in order to find P (O | \( \lambda \)), using an unsophisticated way of calculating it might make it solvable in exponential time. Given T observations and N states, there are \( N^T \) possible state sequences. Even small N and T values, ( say N = 10 and T = 10), contain 10 billion different paths.

Thus, to answer the question of “how” to calculate the P (O | \( \lambda \)), we can use the forward algorithm or backward algorithm to make sure that the computation does not reach exponential time. Given an HMM \( \lambda \), the probability that at time t, the state is i and the partial observation \( o_1, ..., o_t \) has been generated is

\[
\alpha_t(i) = P(o_1, ..., o_t, q_t = S_i | \lambda)
\]

There are three phases involved in the Forward Algorithm

**Initialization**

\[
\alpha_0(i) = \pi_i b_i(0) \quad 1 \leq i \leq N
\]

Where, \( \pi \) denotes the probabilities of initial states

**Induction:**

\[
\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \quad 2 \leq t \leq T, \; i \leq j \leq N
\]

**Termination:**

\[
P(O | \lambda) = \sum_{i=1}^{N} \alpha_T(i)
\]

**Complexity of Forward Algorithm**

Forward Algorithm takes \( N^3T \) computations, unlike the straight forward naïve method, which takes \( (2T \cdot N^T) \) computations.
Solution using Backward Algorithm

Backward Algorithm is just like the Forward Algorithm, but in the opposite direction. Given an HMM $\lambda$, the probability that at time $t$, the state is $i$ and the partial observation $o_{t+1}, \ldots, o_T$ has been generated is

$$
\beta_t(i) = P(o_{t+1}, \ldots, o_T, q_t = S_i | \lambda)
$$

There are three phases involved in the Backward Algorithm

- Initialization: $\beta_t(i) = 1, \quad 1 \leq i \leq N$
- Induction: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad t = T-1 \text{ to } 1, \quad 1 \leq i \leq N$
- Termination: $P(O | \lambda) = \sum_{i=1}^{N} \pi_i \beta_1(i)$

Where, $\pi$ denotes the probabilities of initial states

Problem 2

Given the observation sequence $O$, where $O = o_1, o_2, \ldots, o_t$ and an HMM Model $\lambda$, where $\lambda = (A, B, \pi)$, How do we find the most probable sequence of states that produces the sequence?

This problem is also known as decoding problem.

Solution

This problem can be addressed using Viterbi Algorithm. The Figure 3.1 [8], below explains the various phases of Viterbi algorithm and its calculation.
Viterbi Algorithm

- Initialization ($i=0$): \[ v_0(0) = 1, \quad v_k(0) = 0 \quad \text{for} \quad k > 0 \]

- Recursion ($i=1...L$): \[ v_i(i) = e_i(x_i) \max_k(v_k(i-1)a_{ii}) \]
  \[ \text{ptr}_i(i) = \arg\max_k(v_k(i-1)a_{ii}) \]

- Termination: \[ P(x, \pi^*) = \max_k(v_k(L)a_{i0}) \]
  \[ \pi^*_L = \arg\max_k(v_k(L)a_{i0}) \]

- Traceback ($i=L...1$): \[ \pi^*_{i-1} = \text{ptr}_i(\pi^*_i) \]

Figure 3.1

Viterbi Algorithm is used to track the most probable path of an HMM, given sequence of observations. It uses dynamic programming to calculate the most probable path. For example, consider the following HMM in Figure 3.2 [8].
As per the requirement, we have to find the states that belong to the most probable path, for the sequence \{sunny, cloudy, sunny\}. Let us see how the calculation has been done using Viterbi Algorithm to find the most probable path from the Figure 3.3 [8].
Example of Viterbi algorithm

<table>
<thead>
<tr>
<th>$v$</th>
<th>Sunny</th>
<th>Cloudy</th>
<th>Sunny</th>
</tr>
</thead>
<tbody>
<tr>
<td>(begin) 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Low 0</td>
<td>(0.1)(0.5) - 0.05</td>
<td>0.9 max {(0.05)(0.7)</td>
<td>(0.4)(0.4)} 0.035</td>
</tr>
<tr>
<td>0.16</td>
<td>= (0.9)(0.16) = 0.144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High 0</td>
<td>(0.8)(0.5) = 0.4</td>
<td>0.2 max {(0.05)(0.3)</td>
<td>(0.4)(0.6)} 0.015</td>
</tr>
<tr>
<td>0.24</td>
<td>= (0.2)(0.24) - 0.048</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.8 max {(0.144)(0.3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.0288</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>= (0.8)(0.0432) = 0.03456</td>
</tr>
</tbody>
</table>

Figure 3.3

So, the most probable path is $P(\text{HLH})$.

Problem 3

How do we adjust the HMM to maximize $P(O | \lambda)$?

This problem is known as learning/training problem.

Solution

This problem is same as the question, How do you determine the transition and symbol probabilities of an HMM? The HMM should be initially trained with the data "sufficient enough" to calculate the transition and symbol probabilities. As HMM learns more and more, the efficiency of its output begins to increase.

Graphical representation of learning curve is given in Figure 3.4 [9].
Learning Curve

Figure 3.4
CHAPTER 4

DESIGN OF HMM FOR THE STANDARDIZATION OF REFERENCES

4.1 Purpose of the experiment

At the end of every technical paper, lists of bibliographic citations are listed known as "References". Despite these recommended styles in citations (like MLA), there are citations that come in style that aren’t considered as standards. This occurs because, not all the authors follow the recommended citation style.

There are several formatting tools available to format the citations. One of the most commonly used formatting tools is BibTex [4]. The sources are cited in a consistent manner by making the bibliographic information separate from the presentation information using the BiBTex. In order to enter a record in BibTex, one has to enter each part of the citation separately, Ex: Author, Title, Date, Page Number etc. Extracting the citation parts properly and then separating it into its parts is the main aim of this project.

For example, consider the following citation

Rabiner and Juang, An Introduction to Hidden Markov Models, IEEE, vol. 3, 4-16, 1986

This citation should be parsed and it should extracted the following way

Author  = Rabiner, Juang
Title    = An Introduction to Hidden Markov Model
Journal  = IEEE
Because of the style differences, the citation parts may or may not occur in the same order. Extracting the citations manually from the document is a mammoth task, as citations vary from document to document. The purpose of this project is to design and train the HMM to extract the citations to its parts.

4.2 Technology used

This project was developed using the language VB.Net using Visual Studio 2003. This experiment was run in Windows XP environment.

4.3 Deciding the states

The number of states and their names are the first to be decided before designing the actual HMM. The states cannot be chosen dynamically for creating the HMM. It is the designer who should take the responsibility of identifying the states before the design.

4.3.1 State identification

After analyzing a huge number of citations at the end of the technical papers, a decision was made on the total number of states and the names of the states. There were 14 states totally. The name and purpose of those states are listed down in the Table 4.1
<table>
<thead>
<tr>
<th>S.No</th>
<th>Name</th>
<th>Purpose of the state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start</td>
<td>Start State</td>
</tr>
<tr>
<td>2</td>
<td>Name</td>
<td>To identify the name of the author of the article.</td>
</tr>
<tr>
<td>3</td>
<td>Title</td>
<td>To identify the title of the article.</td>
</tr>
<tr>
<td>4</td>
<td>Journal</td>
<td>Name of the journal where the article was published.</td>
</tr>
<tr>
<td>5</td>
<td>Volume</td>
<td>Volume number of the journal, where the article was published.</td>
</tr>
<tr>
<td>6</td>
<td>Book</td>
<td>Identifies the name of the book</td>
</tr>
<tr>
<td>7</td>
<td>Author</td>
<td>Identifies the name of author of the book</td>
</tr>
<tr>
<td>8</td>
<td>Chapter</td>
<td>The chapter number, which the actual article is written.</td>
</tr>
<tr>
<td>9</td>
<td>Other</td>
<td>Other kind of classification like Technical Report or PhD Thesis.</td>
</tr>
<tr>
<td>10</td>
<td>Affiliation</td>
<td>The industry name or the department name to which the “Other” kind of article belongs to.</td>
</tr>
<tr>
<td>11</td>
<td>Page Number</td>
<td>Page number of the article in the book or journal or report.</td>
</tr>
<tr>
<td>12</td>
<td>City</td>
<td>Name of the city</td>
</tr>
<tr>
<td>13</td>
<td>State</td>
<td>Name of the state</td>
</tr>
<tr>
<td>14</td>
<td>Date</td>
<td>Date of publication of the article</td>
</tr>
<tr>
<td>15</td>
<td>End</td>
<td>End state</td>
</tr>
</tbody>
</table>

Table 4.1

4.4 Deciding the symbols

The symbols that are considered for the designing and the training of HMMs are totally seven in number. They are listed down in Table 4.2
To minimize the complexity of the current project, symbols that belong to (a to z) or (A to Z) are generalized as alphabets. Similarly, any symbols from (0 to 9) are categorized as Numbers.

4.5 Designing the HMM

In order to design the HMM, we need to know the names of the states that will be used and the symbols that each state would emit. Further, the transition probabilities, which are the probability of moving from one state to another, should be available along with the symbol probabilities, which is the probability of each state emitting a symbol. These two probabilities are calculated during the training phase of the HMM.
5.1 Data for HMM

After designing the HMM, the next concern is about the data. Two types of data will be provided for the HMM. They are

a. Training Data

b. Test Data

The training data is the data that will be given to let the program calculate the transition probabilities and Symbol probabilities. Test data is the data that will be given as an input to the program, whose output should be the names of the citation parts extracted.

5.2 Training the HMM

A series of data are given to the HMM for the training purpose. These data should be standardized and tagged before feeding it as a training data [8].

5.2.1 Why standardization?

Non-standardized training data cannot be given to the HMM for training. The data that is going to be used for training should be standardized and tagged first. For example,
in case of address standardization example, the address could be of any of the following format.

No.356, ABC Street, Las Vegas, NV, 89119

#356, S ABC Street, Las Vegas, Nevada, 89119

356, South ABC Street, Las Vegas, NV, 89119

#356, ABC Street, Las Vegas, NV, 89119-1203

This data can be standardized by allowing only one of the above flavors. For example, say we allow the flavor

356, ABC Street, Las Vegas, NV, 89119

So, the training data that does not come in standardized format should be standardized to the format that is specified above. The purpose of standardization is to make it easy for tagging in the later part.

There are several researches about standardization and tagging of the training data. However, our concern is more on training the HMM. So to make it more understandable and simpler as of now, the training data can be standardized and tagged using database or XML tags.

The above standardized data can be tagged as follows using XML

<Data>
  <Number>356</Number>
  <Street Name>ABC</Street Name>
  <City>Las Vegas</City>
  <State>Nevada</State>
  <Zip>89119</Zip>
</Data>
5.2.2 Standardization and tagging in this project

Standardization for the training data is done such that, the symbols that comes in the training data is same as that of the symbols specified in Table 4.2. For instance, in some conditions, the page number may be given as 121:123, where as in some situations it might be given as 121-123. This can be standardized by following only one instance of page number and I followed 121:123 in this project.

Similarly, in some situations Date will be represented as 12-11-2001 or 12-Nov-2001 or Nov-12-2001 or 12/11/2001 etc. In this project, Dates are standardized as DD; MM; YYYY format.

Other condition, that is worthy of mentioning about standardization is the state name. In some citations, the state names are mentioned in short form. For example, “CA” is the abbreviated form for California; “TX” is the abbreviated form for Texas. I have used the full name of the state and not the abbreviated names.

Now let us see about how the training data was tagged in detailed. Figure 5.1 is the snapshot of the training data. The training data is in XML format. The state names are given as the tag names. The tags are arranged based on the order by which the citation parts were arranged in the bibliography of a technical paper. This lets the HMM have its transition probabilities distributed evenly.
The data under the tag `<TrainingData> </TrainingData>` represents a single instance of the training data. The more the training data is, the better the learning would be.

5.3 Calculating the transition probabilities

The most common approach to calculate the state transition probabilities is using the Maximum Likelihood Estimate (MLE).
\[ P(S_i, S_j) = \frac{\text{Number of Transitions from } S_i \text{ to } S_j}{\text{Total Number of Transitions out of } S_i} \]

Consider the following short example, which has two training data.

Name, Title, Journal, Date

Name, Journal, Title, Date

Now the transition probability will be calculated using MLE as follows.

In both the cases, the start state is Name. So the initial probability, which is the probability of the state being a "Name" is 1. From "Name", the next possible state can be either "Title" or "Journal". Total Number of transitions out of "Name" = 2.

\[ P(\text{Name, Title}) = \frac{\text{Number of Transitions from } \text{Name} \text{ to } \text{Title}}{\text{Total Number of Transitions out of } \text{Name}} \]

\[ = \frac{1}{2} \]

\[ = 0.5 \]

\[ P(\text{Name, Journal}) = \frac{\text{Number of Transitions from } \text{Name} \text{ to } \text{Journal}}{\text{Total Number of Transitions out of } \text{Name}} \]

\[ = \frac{1}{2} \]

\[ = 0.5 \]

\[ P(\text{Title, Journal}) = \frac{\text{Number of Transitions from } \text{Title} \text{ to } \text{Journal}}{\text{Total Number of Transitions out of } \text{Title}} \]

\[ = \frac{1}{2} \]

\[ = 0.5 \]

\[ P(\text{Title, Date}) = \frac{\text{Number of Transitions from } \text{Title} \text{ to } \text{Date}}{\text{Total Number of Transitions out of } \text{Title}} \]
The transition probability of the above states can be represented using a matrix.

Table 5.1 shows the matrix that represents the transition probability of the example that was just under discussion.

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Title</th>
<th>Journal</th>
<th>Date</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Title</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Journal</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>Date</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.1
This can be diagrammatically represented as follows in Figure 5.2

In the similar way, code was written such that it calculates the MLE automatically and creates the matrix for all the 15 states (Including the "State" and the "End" states).

5.4 Structure of the test data

The test data can be a raw text. For ease of file access, it is given in an XML file in our project which was improperly tagged. Each and every string formed by tokens or symbols are separated by a delimiter ".".
Figure 5.3 shows the structure of the test data given to the program

```
<References>
  <InputData>
    <Data>Swamy, S.S, H.W. Markov Models, IEEE. Transaction for engineers, 7, 10:12, 12; 1997</Data>
  </InputData>
</References>
```

5.5 Calculating the symbol probabilities

When a test data is given to the program, each and every symbol of the data is parsed by the program and its corresponding symbol probability is calculated and stored in emission probability matrix. Symbol probabilities are calculated as follows

\[ P(w|s) = \frac{\text{No. of times symbol } w \text{ is emitted at state } s}{\text{Total number of times symbol } w \text{ was emitted}} \]

This calculation is same as the calculation that was done for transition probabilities using MLE technique.

Consider the test data

Sam, Introduction to Markov Models, IEEE, 2001

Here, the program first parses through ‘S’ and finds the probability of the symbol ‘S’ being a Name, Title, Journal, etc. Then the symbol ‘a’ is parsed and its probability of being in every state is calculated till the end of the data, that is, till the character ‘l’. Then these probabilities are tabulated in symbol probability matrix.
The actual symbol probability of the HMM is tabulated in Table 5.2. This table will be inputted along with the transition probability matrix to calculate the “alpha” table, using which we can find the most probable path.

Table 5.2 tabulates the probability of every category of the symbols being emitted from a state. That is, if a symbol is an alphabet, then the probability of that alphabet being a Name or Title or Journal is tabulated. This is done for all the 7 categories of symbol. To avoid the least probable symbols having a value of 0, which might affect the overall calculations for a bulky data, a very minimal threshold value is assigned to every symbol pertaining to a state. This process is known as Smoothing.
Symbol Probability of the HMM

This table specifies the actual symbol probabilities.

The symbol V in the table below denotes the value 0.000000001 for smoothing.

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>Jour</th>
<th>Vol</th>
<th>Book</th>
<th>Auth</th>
<th>Chap</th>
<th>Aff</th>
<th>Pg</th>
<th>City</th>
<th>State</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-Z</td>
<td>.24</td>
<td>.34</td>
<td>.16</td>
<td>V</td>
<td>.009</td>
<td>.001</td>
<td>V</td>
<td>.02</td>
<td>V</td>
<td>.1</td>
<td>.06</td>
</tr>
<tr>
<td>0-9</td>
<td>V</td>
<td>.006</td>
<td>.05</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>.02</td>
<td>.3</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>;</td>
<td>.74</td>
<td>.06</td>
<td>.15</td>
<td>V</td>
<td>V</td>
<td>.01</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>:</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>;</td>
<td>V</td>
<td>.88</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>,</td>
<td>.94</td>
<td>.04</td>
<td>.01</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>Space</td>
<td>.12</td>
<td>.47</td>
<td>.17</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>.05</td>
<td>1</td>
<td>.06</td>
<td>.02</td>
</tr>
</tbody>
</table>

Table 5.2
5.6 Calculating the most probable path

First, the "alpha" matrix was calculated using the symbol probabilities and transition probabilities. Then, using the "alpha" matrix, we can retrieve the most probable path. The algorithm that was used to calculate the "alpha" matrix was Forward Chaining Algorithm or Forward Algorithm.

Given below is the pseudo code of the Forward algorithm that was followed in the project [11].

5.6.1 Pseudo code for Forward Chaining Algorithm

```plaintext
# Set of states: Array S
# Start state: s0
# End state: se
# Symbol sequence: Array w
# State transition probabilities: Matrix a
# Symbol emission probabilities: Matrix b
# alpha: Matrix alpha

# All indices in arrays start on 1 in this pseudocode
# Returns total probability: p

# Initialization F1
foreach s in S do
    alpha[1][s] := a[s0][s] _ b[s][w[1]]
done

# Induction F2
for i := 1 to length(w) - 1 do
```
foreach s in S do
    foreach s0 in S do
        alpha[i + 1][s] += alpha[i][s0] * a[s0][s]
    done
    alpha[i + 1][s] = b[s][w[i + 1]]
    done
# Termination F3
foreach s in S do
    p += alpha[length(w)][s] * a[s][se]
    done
return p

5.6.2 Code snippet of Forward Chaining Algorithm

'Construct the first row - Initialization Phase
    For i As Integer = 1 To 14
        Result(1, i) = TransProb(1, i) * EmisProb(i, GetEmisIndex(chrSymbolSeq(0)))
    Next

'Construct the remaining rows - Induction Phase
    For i As Integer = 0 To chrSymbolSeq. Length - 2
        For j As Integer = 1 To 13
            For k As Integer = 1 To 13
                Result(i + 1, j) += (Result(i, k) * TransProb(k, j))
            Next
            Result(i + 1, j) *= EmisProb(j, GetEmisIndex(chrSymbolSeq(i + 1)))
        Next
    Next

'Construct the last row - Termination Phase
    For i As Integer = 1 To 14
        Result(chrSymbolSeq. Length, i) += Result(chrSymbolSeq. Length - 1, i) * TransProb(i, 14)
p += Result(chrSymbolSeq. Length - 1, i) * TransProb(i, 14)
    Next
5.6.3 Calculating the output

After calculating the “alpha” matrix, the program calls a function called “displaypath()”, which gets the length of each individual string up to the next delimiter, uses that further to read through the “alpha” matrix and finds the probability of every state pertaining to the string length.

For example, the first string “Swamy”, has the length of 5. After calculating the alpha matrix, the probability of all the 14 columns of the 5th row is read and the most probable cell signifies the most probable state. In our case, if the cell (5, 1) has the maximum value comparing to the other (5, x) values where x >= 2 and x<15, then the most probable state to which the string “Swamy” belongs to is “Name”. If the value of (5,2) is high, then the state that most probably emits that string is “Title”.

This calculation is done by the function “displaypath()” in the program and it outputs the path name in the string format in a VB Message box.

5.6.4 Output screen:

The snapshot of the output screen is provided in Figure 5.4. The input data or the test data is same as that of Figure 5.3
Figure 5.4
6.1 Terminologies

Before calculating the efficiency, let us have a brief look over the terminologies that are involved in calculating the efficiency of the HMM.

6.1.1 True Positive (TP)

A true positive (TP) is an instance where the HMM finds a tagged state in the correct place. For example, if a “Name” instance of a test data is correctly identified of “Name”, then it is a TP.

6.1.2 False Negative (FN)

A false negative (FN) is an instance where the HMM didn’t find a state. For example, if the instance “Name” was not found by the HMM in the test data when it was supposed to be found.

6.1.3 False Positive (FP)

A false positive (FP) is an instance where the HMM thought it found a state, where there wasn't one.

6.1.4 Precision

In IR, precision is the ratio of documents that are relevant to the user’s query to the retrieved documents [1].
In our scenario, precision can be calculated by the following formula

\[
\text{Precision} = \frac{\text{True positive}}{\text{True Positive} + \text{False Positive}}
\]

6.1.5 Recall

In IR, Recall is the fraction of the documents that are relevant to the query that are successfully retrieved [1].

\[
\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}
\]

In our scenario, recall can be calculated by the following formula

\[
\text{Recall} = \frac{\text{True positive}}{\text{True Positive} + \text{False Negative}}
\]

6.1.6 F-Measure

It is the weighted harmonic mean of precision and recall [1]. This is also known as F1. F1 can be calculated by the formula

\[
F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.
\]

A total of 150 data were given for training and 80 data were given for testing. The results obtained are in response to these data. Results were noted for a sample of 50, 60, 70 and 80 data. Over-all Precision, Recall and F-Measure were found and tabulated in the table 6.1. This is the final result of the project.

Since the number of training data was more towards finding the path which has the states Name, Title, Journal, Volume, Page Number, State and Date, the precision, recall and F-Measure were calculated for these states.
<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>Journal</th>
<th>Volume</th>
<th>Page Number</th>
<th>State</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.77</td>
<td>1.0</td>
<td>0.92</td>
<td>0.67</td>
<td>0.77</td>
<td>0.69</td>
</tr>
<tr>
<td>Precision</td>
<td>0.89</td>
<td>0.69</td>
<td>0.85</td>
<td>0.89</td>
<td>0.78</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Table 6.1
CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

Table 6.1 shows the actual results that were obtained through the experiment. The results of this experiment come closer to the best results which are generally around 0.8 or 0.9. There are several areas, where this experiment can be improved which are now reserved for the future works. In the initial training of the HMM, we found out various training data which had unpredictable order. The 80 references used were enough to get a reasonable result. However, a larger dataset is required for an improved performance in the future. Figure 7.1 shows the partial diagrammatic representation of the HMM after the training.

Figure 7.1
7.2 Future works

In the future, this project can be improved by

1. Providing more dataset for the training.
2. Adding more number of states.
3. Instead of restricting the symbol categories into seven, we can let each alphabet (both upper and lower case) and number to have its probability with more number of special characters. By doing so, there will be an hike in the performance.
BIBLIOGRAPHY


APPENDIX I

TRAINHMM.VB

Imports System
Imports System.Xml
Public Class TrainHMM
    Dim LookupTable As DataTable
    Dim TransitionProb(14, 14) As Double
    Dim countVal(14, 14) As Integer
    Dim volcnt As Integer = 0
    Dim pgcnt As Integer = 0
    Dim statecnt As Integer = 0
    Dim datecnt As Integer = 0
    Dim idCount(16) As Integer
    Dim keepOrder As String

    ' This constructor Initializes all the objects of type Node.
    ' These Objects are considered as the states available in the HMM
    Public Sub New()
        Call Initialize()
    End Sub

    ' Initializing ObjectIds Explicitly
    Private Sub Initialize()
        For i As Integer = 0 To 14
            idCount(i) = 0
        Next

        ' Initialize the Transition Probability Matrix
        For i As Integer = 0 To 14
            For j As Integer = 0 To 14
                TransitionProb(i, j) = 0.0
            Next
        Next

        For i As Integer = 0 To 14
            For j As Integer = 0 To 14
                countVal(i, j) = 0
            Next
        Next

        ' The columns in the Lookup Table is created
        ' This will further be used in the program to get populated
        Private Sub InitializeLookUp()
            LookupTable = New DataTable
        End Sub

45
LookupTable.Columns.Add("Name")
LookupTable.Columns.Add("Title")
LookupTable.Columns.Add("Journal")
LookupTable.Columns.Add("Volume")
LookupTable.Columns.Add("Book")
LookupTable.Columns.Add("AuthorName")
LookupTable.Columns.Add("Chapter")
LookupTable.Columns.Add("Other")
LookupTable.Columns.Add("Affiliation")
LookupTable.Columns.Add("PageNumber")
LookupTable.Columns.Add("City")
LookupTable.Columns.Add("State")
LookupTable.Columns.Add("Date")

End Sub
' The trained values of HMM will be stored in the file along with the size of the training data file.
' If there is an increase in the size of the training data and if needs to be trained then it returns true
' If not, it returns false
Private Function ShouldTrain() As Boolean

End Function
' This function reads the data from the XML Training Data and feeds it to the lookup Table
' This function returns the lookup table
Private Function StoreLookUp() As DataTable

' Declaring the row string
Dim rowstring As String
Dim readString As String

' Declaring the row items
Dim Name As String
Dim Title As String
Dim Journal As String
Dim Volume As String
Dim Book As String
Dim Author As String
Dim Chapter As String
Dim Other As String
Dim Affiliation As String
Dim Page As String
Dim City As String
Dim State As String
Dim [Date] As String

Dim charcnt As Integer = 0
Dim chararr As Char()
Dim iter As Integer = 0

' Initializes the Lookup Table
Call InitializeLookUp()
Dim xmlReader As New XmlTextReader("C:\Documents and Settings\Owner.YOUR-32F8E4BF84\My Documents\Visual Studio Projects\HMM\XMLFile1.xml")
Dim caseStr As String = ""

46
Dim arrLookup As String()
While xmlReader.Read()
    iter += 1
    caseStr = xmlReader.Name
    Select Case caseStr
        Case "Trainingdata"
            If keepOrder <> "" Then
                AssignProbablity(keepOrder, iter)
            End If
            If Name <> "" Then
                LookupTable.Rows.Add(New Object() {Name, Title, Journal, Volume, Book, Author, Chapter, Other, Affiliation, Page, City, State, [Date]})
            End If
            charcnt = 0
            'Clearing all datas
            Name = ""
            Title = ""
            Journal = ""
            Volume = ""
            Book = ""
            Author = ""
            Chapter = ""
            Other = ""
            Affiliation = ""
            Page = ""
            City = ""
            State = ""
            [Date] = ""
            keepOrder = ""
        Case "Name"
            Name = xmlReader.ReadString()
            'MsgBox(Name)
            chararr = Name
            If Name <> "" Then
                For i As Integer = 0 To Name.Length - 1
                    If chararr(i) = "","c Then
                        charcnt += 1
                    End If
                Next
                For i As Integer = 0 To charcnt
                    If keepOrder = "" Then
                        keepOrder = "1"
                    Else
                        keepOrder = keepOrder + "," + "1"
                    End If
                Next
            End If
        Case "Title"
        'Code for processing "Title"
Title = xmlReader.ReadString()

If Title <> "" Then
    If keepOrder = "" Then
        keepOrder = "2"
    Else
        keepOrder = keepOrder + "," + "2"
    End If
End If

Case "JName"
    Journal = xmlReader.ReadString()

    If Journal <> "" Then
        If keepOrder = "" Then
            keepOrder = "3"
        Else
            keepOrder = keepOrder + "," + "3"
        End If
    End If
End If

Case "BName"
    Book = xmlReader.ReadString()

    If Book <> "" Then
        If keepOrder = "" Then
            keepOrder = "5"
        Else
            keepOrder = keepOrder + "," + "5"
        End If
    End If
End If

Case "OName"
    Other = xmlReader.ReadString()

    If Other <> "" Then
        If keepOrder = "" Then
            keepOrder = "8"
        Else
            keepOrder = keepOrder + "," + "8"
        End If
    End If
End If

Case "JVol"
    Volume = xmlReader.ReadString()

    If Volume <> "" Then
        If keepOrder = "" Then
            keepOrder = "4"
        Else
            keepOrder = keepOrder + "," + "4"
        End If
    End If
End If

Case "Author"
    Author = xmlReader.ReadString()

    If Author <> "" Then
        If keepOrder = "" Then
            keepOrder = "6"
        Else
            keepOrder = keepOrder + "," + "6"
        End If
    End If
End If
keepOrder = keepOrder + "," + "6"
End If
End If
Case "Chapter"
    Chapter = xmlReader.ReadString()
    If Chapter <> "" Then
        If keepOrder = "" Then
            keepOrder = "7"
        Else
            keepOrder = keepOrder + "," + "7"
        End If
    End If
End If
Case "Affiliation"
    Affiliation = xmlReader.ReadString()
    If Affiliation <> "" Then
        If keepOrder = "" Then
            keepOrder = "9"
        Else
            keepOrder = keepOrder + "," + "9"
        End If
    End If
End If
Case "City"
    City = xmlReader.ReadString()
    If City <> "" Then
        If keepOrder = "" Then
            keepOrder = "11"
        Else
            keepOrder = keepOrder + "," + "11"
        End If
    End If
End If
Case "State"
    State = xmlReader.ReadString()
    If State <> "" Then
        If keepOrder = "" Then
            keepOrder = "12"
        Else
            keepOrder = keepOrder + "," + "12"
        End If
    End If
End If
Case "Page"
    Page = xmlReader.ReadString()
    If Page <> "" Then
        If keepOrder = "" Then
            keepOrder = "10"
        Else
            keepOrder = keepOrder + "," + "10"
        End If
    End If
End If
Case "Day"
    readString = xmlReader.ReadString()
If readString <> " " Then
    If [Date] <> " " Then
        [Date] = [Date] + readString + ";"
    Else
        [Date] = readString
    End If
End If
Case "Month"
    readString = xmlReader.ReadString()
    If readString <> " " Then
        If [Date] <> " " Then
            [Date] = [Date] + readString
        Else
            [Date] = readString
        End If
    End If
End If
Case "Year"
    readString = xmlReader.ReadString()
    If readString <> " " Then
        If [Date] <> " " Then
            [Date] = [Date] + readString
        Else
            [Date] = readString
        End If
    End If
End If
If [Date] <> " " Then
    If keepOrder = " " Then
        keepOrder = "13"
    Else
        keepOrder = keepOrder + "," + "13"
    End If
End If
End Select
End While
Return LookupTable
End Function
' This function stores all the states inside the hashtable (which is our HMM) and returns the same.
' This function further calls the function AssignProbability() to assign the transition probability.
' Before returning the HMM, write the details of the HMM and the object state into a file.
Private Function CreateHMM() As Hashtable

End Function
' Assigns the transition probability
Private Sub AssignProbability(ByVal order As String, ByVal iter As Integer)
    'MsgBox(order.ToString)
    Dim TransitionProb(16, 16) As Double
    Dim index As Integer = 0
    Dim chararrOrder(order.Length) As Char
    Dim start_from As Integer = 1
    chararrOrder = order
    Dim ch As Char
For i As Integer = 0 To order.Length - 1
    ch = chararrOrder(i)
    If ch = ","c Then
        index += 1
    End If
Next
' MsgBox("Index" + index.ToString)
Dim calc_from As Integer
Dim calc_to As Integer
Dim intarrOrder(index + 1) As Integer
Dim newOrder(order.Length) As Integer

Dim totalCount As Integer = 0
Dim isFirst As Boolean = False

Dim prob_start As Double
Dim prob_next As Double
Dim divisor As Integer = 1
Dim str As String = ""
Dim c As Char
Dim cnt As Integer = 1
Dim startint As Integer = 0
Dim endint As Integer = 0

chararrOrder = order
'Put into an Integer array
For i As Integer = 0 To order.Length - 1
    c = chararrOrder(i)
    If c <> "," Then
        str = str + c.ToString
        If i = order.Length - 1 Then
            intarrOrder(cnt) = Integer.Parse(str)
            str = ""
        End If
    Else
        intarrOrder(cnt) = Integer.Parse(str)
        str = ""
        cnt += 1
    End If
Next
Dim startintCnt As Integer = 0
cnt = 0
startint = 0
endint = 0

For i As Integer = 1 To intarrOrder.Length - 1
    If i = 1 Then
        endint = intarrOrder(i)
        countVal(0, endint) += 1
        startint = endint
    ElseIf (startint = intarrOrder(i)) And (i <> 1) Then
        startintCnt += 1
    End If
Next
Else
    countVal(startint, endint) += startintCnt
    startintCnt = 0
    endint = intarrOrder(i)
    countVal(startint, endint) += 1
    startint = endint
End If

' Storing the end state.
If i = intarrOrder.Length - 1 Then
    countVal(intarrOrder(i), 14) += 1
End If
Next

End Sub

' The primary function that calls the other functions in this class
to train the HMM.
' This function checks the value of ShouldTrain function.
' If it returns 'true', then the training takes place, else the
values are stored from the file.
' Then the HMM and the Lookuptable will be returned in a HashTable

Public Function ProvideTrainedHMM() As Hashtable
    Dim HMM As New Hashtable
    Dim total As Integer = 0

    Dim key As Integer = 0
    LookupTable = StoreLookUp()

    ' Store the transition probability
    For i As Integer = 0 To 14
        For j As Integer = 0 To 14
            total = total + countVal(i, j)
        Next
        For j As Integer = 0 To 14
            If total > 0 Then
                TransitionProb(i, j) = (countVal(i, j) / total)
            End If
        Next
        total = 0
    Next

    ' Checking the Lookup Table
    For i As Integer = 0 To LookupTable.Rows.Count - 1
        For j As Integer = 0 To LookupTable.Columns.Count - 1
            Dim str As String = ""
            str = CType(LookupTable.Rows(i).Item(j), String)
        Next
    Next

    ' Storing hashtable
    HMM.Add(key, TransitionProb)
    key += 1
    HMM.Add(key, LookupTable)
Return HMM
End Function
End Class
Imports System
Imports System.Xml

' This class is responsible for generating the alpha matrix and then
' calculating the most probable path.
Public Class Viterbi
    Dim TransProb(14, 14) As Double
    Dim EmisProb(14, 8) As Double
    Dim EmisCount(14, 8) As Integer
    Dim LookupTable As New DataTable
    Dim colsize As Integer = 0
    Dim trip As Integer = 0
    Dim FinalRow(15) As Double
    Dim FinalPath As String = ""
    ' Can take a value from 0 to 6
    ' 0 - Begin
    ' 1 - Alphabet
    ' 2 - Number
    ' 3 - .
    ' 4 - :
    ' 5 - ;
    ' 6 - ,
    ' 7 - ""
    Dim PrevSymbol As Byte = 0
    Dim startlndex As Integer = 0
    Dim endlndex As Integer = 0
    ' This function is generally used to initialize an array
    Private Sub Initialize(ByRef intArr() As Integer)
        For i As Integer = 0 To 14
            For j As Integer = 0 To 7
                intArr(i, j) = 0
            Next
        Next
    End Sub
    ' Calculates and stores the emission probability
    Private Sub StoreCount(ByVal p_SymSeq As String)
        p_SymSeq = p_SymSeq.ToLower
        Dim chrSymbolSeq() As Char = p_SymSeq
        Dim total As Integer = 0
        Dim retValue As Integer = 0
        Dim matrixRow As Integer = 0
        Dim matrixCol As Integer = 0
        Dim str As String = ""
        For i As Integer = 0 To chrSymbolSeq.Length - 1
            ...
For c As Integer = 0 To LookupTable.Columns.Count - 1
    matrixRow = c + 1
    For r As Integer = 0 To LookupTable.Rows.Count - 1
        str = CType(LookupTable.Rows(r).Item(c), String)
        str = str.ToLower
        retval = str.IndexOf(chrSymbolSeq(i))
        If retval >= 0 Then
            If chrSymbolSeq(i) Like "]\[A-Z|[a-z]\[" Then
                matrixCol = 1
            ElseIf chrSymbolSeq(i) Like "]\[0-9\[" Then
                matrixCol = 2
            ElseIf chrSymbolSeq(i) = "]\[.c Then
                matrixCol = 3
            ElseIf chrSymbolSeq(i) = "]\[:c Then
                matrixCol = 4
            ElseIf chrSymbolSeq(i) = "]\[;c Then
                matrixCol = 5
            ElseIf chrSymbolSeq(i) = "]\[,c Then
                matrixCol = 6
            ElseIf chrSymbolSeq(i) = "]\[c Then
                matrixCol = 7
            End If
            EmisCount(matrixRow, matrixCol) += 1
        End If
    Next
Next
Next
Next
' Store the emission probability
For j As Integer = 0 To 7
    For i As Integer = 1 To 14
        total = total + EmisCount(i, j)
    Next
For i As Integer = 1 To 14
    If total > 0 Then
        EmisProb(i, j) = (EmisCount(i, j) / total)
    End If
    If EmisProb(i, j) = 0.0 Then
        EmisProb(i, j) = 0.00000000001
    End If
Next
EmisProb(0, 1) = 1.0

Next
End Sub

' This routine is responsible for calculating the emission probability
Private Sub EmissionProb() 
    Dim xmlReader As New XmlTextReader("C:\Documents and Settings\Owner.YOUR-32F8E4EF84\My Documents\Visual Studio Projects\HMM\XMLFile2.xml")
    Dim casestr As String
    Dim strSymbolSeq As String
    Dim total As Integer

While xmlReader.Read()
    casestr = xmlReader.Name

    If casestr.Length > 4 Then
        If (casestr.Substring(0, 4)).Equals("Data") Then
            casestr = "Data"
        End If
    End If
End If

Select Case casestr
    Case "Inputdata"

        Case "Data"
            strSymbolSeq = xmlReader.ReadString
            If colsize < strSymbolSeq.Length Then
                colsize = strSymbolSeq.Length
            End If
            StoreCount(strSymbolSeq)
    End Select
End While

End Sub

' This function assigns index for every symbol
Private Function GetEmisIndex(ByVal c As Char) As Integer

    Dim index As Integer = 0
    If c Like "[A-Z|a-z]" Then
        index = 1
    ElseIf c Like "[0-9]" Then
        index = 2
    ElseIf c = "." Then
        index = 3
    ElseIf c = ":" Then
        index = 4
    ElseIf c = "," Then
        index = 5
    ElseIf c = " " Then
        index = 6
    ElseIf c = " " Then
        index = 7
    End If

    Return index
End Function

' Forward Chaining Algorithm Implementation
Private Function ForwardChaining() As Double

    Dim Result(colsize, 14) As Double
    Dim p As Double = 0.0

    ' Initialize to zero
    For i As Integer = 0 To 7
        For j As Integer = 0 To 14
            Result(i, j) = 0
        Next
    Next

    ' Forward Chaining Algorithm
    For i As Integer = 0 To 7
        For j As Integer = 0 To 14
            For k As Integer = 0 To 14
                Result(i, j) = Result(i, j) + Result(i, k) * p
            Next
            p = p + 1
        Next
    Next

    Return p
End Function
result(i, j) = 0.0
Next
Next

' Reading the string and then calculating the resultant matrix and total probability
Dim xmlReader As New XmlTextReader("C:\Documents and Settings\Owner.YOUR-32F8E4EF84\My Documents\Visual Studio Projects\HMM\XMLFile2.xml")
Dim casestr As String
Dim strSymbolSeq As String
Dim total As Integer
Dim chrSymbolSeq() As Char
Dim count As Integer = 0
Dim inputData() As String
Dim len As Integer
Dim ind As Integer = 3
Dim base As Integer = 10

While xmlReader.Read()
    Dim ht As Hashtable
    casestr = xmlReader.Name
    If casestr.Length > 4 Then
        If (casestr.Substring(0, 4)).Equals("Data") Then
            casestr = "Data"
        End If
    End If
    Select Case casestr
        Case "Inputdata"
        Case "Data"
            count += 1
            strSymbolSeq = xmlReader.ReadString
            chrSymbolSeq = strSymbolSeq
            trip = count
            inputData = strSymbolSeq.Split("c")
            len = inputData(0).Length + inputData(1).Length + inputData(2).Length + 2

            'Construct the first row - Initialization Phase
            For i As Integer = 1 To 14
                Result(1, i) = TransProb(1, i) * EmisProb(i, GetEmisIndex(chrSymbolSeq(0)))
            Next

            'Construct the remaining rows - Induction Phase
            For i As Integer = 0 To chrSymbolSeq.Length - 2
                For j As Integer = 1 To 13
                    For k As Integer = 1 To 13
                        Result(i + 1, j) += (Result(i, k) * TransProb(k, j))
                    Next
                Next
            Next
Result(i + 1, j) *= EmisProb(j, GetEmisIndex(chrSymbolSeq(i + 1)))
Next
Next

'Calculate the total Probability - Termination Phase
Dim targetIndex As Integer = 0
Dim lastRow(15) As Double
For i As Integer = 0 To InputData.Length - 1
  If i = 0 Then
    targetIndex = inputData(i).Length
  ElseIf i >= 2 Then
    targetIndex += inputData(i).Length + 1
  Else
    targetIndex += inputData(i).Length
  End If
  For j As Integer = 1 To 13
    lastRow(j) = Result(targetIndex, j)
  Next
  FinalRow = lastRow
  If FinalPath <> "" Then
    FinalPath = FinalPath + "," + displayPath() + "+" + inputData(i)
  Else
    FinalPath = displayPath() + "+" + 
  End If
Next
MsgBox(FinalPath)

'Termination Phase
For i As Integer = 1 To 14
  Result(chrSymbolSeq.Length, i) +=
  Result(chrSymbolSeq.Length - 1, i) * TransProb(i, 14)
  p += Result(chrSymbolSeq.Length - 1, i) * TransProb(i, 14)
Next

End Select
End While
Return p
End Function

#Region "Path Display"
' This Function displays the most probable path
Public Function displayPath() As String
  Dim nextNode As String = ""
  Dim weights As Double = 0.0
  Dim keepIndex As Integer = 0
  Dim maxVal As Double = 0.0
  Dim Val As Double = 0.0
maxVal = FinalRow(0)
For j As Integer = 1 To 14
    If maxVal > FinalRow(j) Then
        Else
            maxVal = FinalRow(j)
            keepIndex = j
            End If
        Next

Select Case keepIndex
    Case 1
        nextNode = "Name"
    Case 2
        nextNode = "Title"
    Case 3
        nextNode = "Journal"
    Case 4
        nextNode = "Volume"
    Case 5
        nextNode = "Book"
    Case 6
        nextNode = "Author"
    Case 7
        nextNode = "Chapter"
    Case 8
        nextNode = "Other"
    Case 9
        nextNode = "Affiliation"
    Case 10
        nextNode = "Page Number"
    Case 11
        nextNode = "City"
    Case 12
        nextNode = "State"
    Case 13
        nextNode = "Date"
End Select
Return nextNode
End Function

'd This function calculates the total probability
Public Function CalculateTotalProb(ByVal p_TransProb As Double(), ByVal p_dt As DataTable)
Dim totalProbability As Double = 0.0
' Get the Symbol Emission Probability
Call Initialize(EmisCount)
EmisCount(0, 1) = 1
' Get the Transition Probability
TransProb = p_TransProb
'smoothing
For i As Integer = 0 To 14
    For j As Integer = 0 To 13

59
If TransProb(i, j) = 0.0 Then
    TransProb(i, j) = 0.0
End If
Next
Next
LookupTable = p_dt
Call EmissionProb()
' Calculate the total probability using Forward Chaining Algorithm
    totalProbability = ForwardChaining()
End Function
End Class
VITA

Graduate College
University of Nevada, Las Vegas

Swamynathan Sambamurthy

Home Address:
4340, Escondido Street
Apt # 3
Las Vegas, Nevada 89119

Degrees:
Bachelor of Technology (IT), 2004
University of Madras, India

Thesis Title: Standardization of References using Hidden Markov Models

Thesis Examination Committee:
Chairperson, Dr. Kazem Taghva, Ph.D.
Committee Member, Dr. Thomas Nartker, Ph.D.
Committee Member, Dr. Yoohwan Kim, Ph.D.
Graduate Faculty Representative, Dr. Venkatesan Muthukumar, Ph.D.