12-1-2012

Post Processing of Optically Recognized Text using First Order Hidden Markov Model

Spandana Malreddy
University of Nevada, Las Vegas, spandana.4160@gmail.com

POST PROCESSING OF OPTICALLY RECOGNIZED TEXT USING FIRST ORDER HIDDEN MARKOV MODEL

by

Spandana Malreddy

Bachelor of Engineering, Computer Science
Jawaharlal Nehru Technological University, India
2009

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

School of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
December 2012

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Spandana Malreddy

entitled

Post Processing of Optically Recognized Text Using First Order Hidden Markov Model

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
School of Computer Science

Kazem Taghva, Committee Chair

Ajoy K. Datta, Committee Member

Laxmi P. Gewali, Committee Member

Venkatesan Muthukumar, Graduate College Representative

Thomas Piechota., Interim Vice President for Research and Graduate Studies
and Dean of the Graduate College

December 2012

ABSTRACT

Post Processing of Optically Recognized Text Using First Order Hidden Markov Model

by

Spandana Malreddy

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

In this thesis, we report on our design and implementation of a post processing system for Optically Recognized text. The system is based on first order Hidden Markov Model (HMM). The Maximum Likelihood algorithm is used to train the system with over 150 thousand characters. The system is also tested on a file containing 5688 characters. The percentage of errors detected and corrected is 11.76% with a recall of 10.16% and precision of 100%

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Dr. Kazem Taghva for his excellent guidance, patience and invaluable support throughout my thesis work. I am grateful to my graduate coordinator, Dr. Ajoy K Datta for his support and help during my masters. I extend my gratitude to my thesis committee members Dr. Laxmi P. Gewali, Dr.Ajoy K Datta and Dr. Venkatesan Muthukumar for their encouragement and insightful comments. I would also like to thank staff of school of computer science for their assistance during my masters.

My special gratitude goes to my family and friends who have always supported, encouraged and believed in me in all my endeavors.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This thesis discusses Hidden Markov Model (HMM) to correct some of the Optical Character Recognition (OCR) errors. Hidden Markov Model consists of non-deterministic process – markov chain. In 1906, a Russian mathematician named Andrey Andreyevich Markov introduced the markov chain while reporting results for a stochastic process [1]. Markov chain defines states which are hidden and the state transition probabilities. Markov process is estimating the state probability at a particular time depending on the state probability in the present. Later in 1960's Leonard E. Baum and others described Hidden Markov Model in the series of statistical papers. "Hidden Markov Model is a statistical Markov Model in which the system being modeled is assumed to be markov process with unobserved (hidden) states" [3]. The output of HMM is the most likely sequence of states with the given observations and other parameters. In 1970's the first application of HMM is Speech Recognition [3] and later in 1980's they were used in biological sequences [3] and then extends its applications to natural language modeling [1], gene predictions etc.,

In this thesis we will discuss about the two algorithms used in HMM which are Viterbi Algorithm and Maximum Likelihood Algorithm (MLE). During the teaching of information theory, Andrew Viterbi developed the Viterbi Algorithm in 1967. It helped him to prove an asymptotically optimum upper bound on error probability of error correcting codes rather than done on sequential codes. Viterbi Algorithm is used to find the most likely sequence path of the hidden states known as Viterbi path. The theory of

MLE algorithm was developed for Bayesian statistics and then later simplified by other authors. Between 1912 and 1922, R. A. Fisher popularized the Maximum Likelihood Estimation algorithm. When we give a set of data as training data to a model, MLE algorithm is used to get the highest probabilities for the model parameters.

OCR text is the text generated when text is scanned using OCR device. OCR device translate the handwritten or the printed text to data that will be in editable format. During this transition, the challenge is to translate the text which is in different format, style to the original text without errors. Kazem, Julie and Allen state that the initial step, scanning of text using OCR device is done by the optical scanner which uses the light intensity to characterize the grey levels and builds a matrix of 1's (for black) and 0's (for white) which represents a bit- mapped image [10]. In this process some of the characters are misrecognized causing the OCR errors. During the process of scanning, some of the characters are deleted, some of characters are inserted and some of them are wrongly recognized. The main idea is to consider HMM for correcting OCR errors as the output of HMM is the most likely sequence path of states when we give a set of characters which is the OCR text as training data.

1.1 Thesis Overview

HMM is used to correct the OCR error data. In our thesis OCR text is taken, trained with the HMM with some initial probabilities and then tested with the different set of text. The resulted output is the correction of text of OCR error data. Output is measured using recall and precision.

Chapter 2 provides a background of HMM. The 3 problems training, decoding and evaluation are explained in detail. Post processing and pre processing of data are also discussed .Chapter 3 gives a detailed explanation of HMM design and implementation. MLE (Maximum Likelihood Algorithm) and Viterbi Algorithm are discussed in this chapter. In chapter 4 we report our experiment results on various data using training file and test file. Results are reported using standard measures: recall and precision. Chapter 5 concludes the thesis and also discusses the way to improve the performance of HMM including the future work.

CHAPTER 2

BACKGROUND

The design and implementation details of first order Hidden Markov Model (HMM) for optically recognized text are discussed in this thesis. The name "Hidden" is given to HMM as the states are hidden and the observations are visible. In first order HMM, the transition probabilities depends only on the previous state but not on the history of the process. HMM is defined by a set of five variables.

The elements of HMM and their notation are discussed in detail in the following section.

2.1 Elements of HMM

HMM is characterized by defining it with 5 variable tuple (S, V, π, A, B).

i. S is the number of states in a model. S= $\{s_1, s_2, s_3, s_4 \ldots s_n\}$. $s_1, s_2, s_3, s_4 \ldots s_n$ are the hidden states of model.

ii. V is the observation symbols. V = $\{v_1, v_2, v_3 \ldots v_n\}$. $v_1, v_2, v_3 \ldots v_n$ are the observation symbols of the states.

iii. π is the initial state probabilities. $\pi = \{ \pi_i \}$. Where

$$\pi_i = p[q_1 = S_i] , 1 \leq i \leq N$$

The initial state probabilities are the probabilities of states at time t = 1.

iv. A is the probabilities of state transitions.

$$A = \{ a_{ij} \}, \text{ where}$$

$$a_{ij} = P(q_{t+1} = S_j \,|\, q_t = S_i), i \geq 1 \text{ and } j \leq N$$

a_{ij} denotes the probability of state S_i going to state S_j at a time t. Transition probabilities are independent of time. A denotes the array of state transition probabilities.

v. B is the emission probabilities

$$B = \{b_j(k)\}, \text{ where}$$

$$b_j(k) = P\ (v_k \text{ at } t \mid q_t = s_j) \quad 1 \leq j \leq N \text{ and } 1 \leq k \leq M$$

$b_j(k)$ denotes the probability of symbol v_k when state s_j is entered at time t. Emission probabilities are also independent of time t.

Generally the model parameters of HMM are denoted as $\lambda = (A, B, \pi)$

2.2 Three problems of HMM

HMM to be used in the real world application deals with three problems. These three problems are addressed based on the observation sequence and state sequence where

$$O = \{o_1, o_2, o_3 \ldots o_t\}$$

$$X = \{x_1, x_2, x_3 \ldots x_t\}$$

O is the Observation sequence. Given the observation sequence, HMM calculates the most possible state sequence (X) based on this observation sequence. The three problems are

- **Evaluation Problem**
- **Training Problem**
- **Decoding problem**

These three problems are discussed in detail:

1. The evaluation Problem is used to find the probability of observations when the model parameters λ and observation set are given. Forward Algorithm is used to solve this problem.
2. The training problem is mainly used to adjust the model parameters of HMM which gives the maximum probability of observation sequence. The input for this problem is the tag file consisting of states and symbols. The training process is important process of the system. When the system is trained with more data, the probabilities are recorded and the system works more accurate. Generally there are two ways of training data: supervised and unsupervised. In supervised training, Maximum Likelihood Estimation algorithm is used where the model is trained with the tag file with the states and corresponding symbols. Unsupervised training is done using Baum-Welch algorithm. Supervised training is used in this thesis.
3. The decoding problem is used to find most probable sequence of states when we give observation sequence and model parameters λ. Viterbi algorithm is used to solve this problem.

Evaluation problem is not addressed in this thesis. Training problem and decoding problem use a lot of dynamic programming and are discussed in detail in this thesis.

We will now discuss the details of viterbi algorithm and Maximum Likelihood Estimation algorithm. These algorithms are explained using the mathematical equations and elements of HMM.

2.3 Viterbi Algorithm

Viterbi algorithm is used to identify the hidden states of the model. Given observation sequence and model parameters, it helps to identify the most probable path for the states. Generally the most probable path is known as viterbi path. It is the best technique and involves a lot of dynamic programming to solve the decoding problem. It is the most used algorithm in recent cases. Viterbi algorithm uses trellis diagram for back tracking the path.

Inputs:

(a) States, S = { $s_1, s_2, s_3, \dots, s_n$ }

(b) Observation symbols, O = {$o_1, o_2, o_3 \dots, o_k$}

(c) Transition matrix A_{ij}. A_{ij} stores the probability of transiting state s_i to s_j.

(d) Emission matrix B_{ik}. B_{ik} stores the probability of observation ok from state s_i.

(e) Initial probabilities π_i where π_i stores the initial probabilities of state s_i.

(f) Observation Sequence Y = {$y_1, y_2, y_3, \dots, y_t$}

Outputs:

$$\delta_n(i) = \max_{s1,s2,s3,\dots,sn} p(s_1, s_2, s_3, \dots, s_n = si, y_1, y_2, y_3, \dots, y_t \mid O)$$

$$\psi_n(i) = \text{argmax}_{s1,\, s2,\, \dots sn}\ p(s_1, s_2, s_3, \dots, s_n = si, y_1, y_2, y_3, \dots, y_t \mid O)$$

Steps in the algorithm:

(i) Initialization

$$\delta_i(1) = \pi_i \,.\, b_{i\, y1}\ , \text{ where } i = 1, 2, 3, \dots N$$

$$\psi_1(i) = 0$$

(ii) Recursion

$\delta_n(j) = \max_{1 \le i \le N} (\delta_{n-1}(i).\ a_{ij})\ .\ b_j\ y_n$, where i = 2,3, … N and j = 1, 2, 3, … N

$\psi_n(j) = \max_{1 \le i \le N} (\delta_{n-1}(i).\ a_{ij})$, where i = 2,3, … N and j = 1, 2, 3, … N

(iii) Termination

$P^* = \max_{1 \le i \le N} \delta_N(i)$

$q^* = \text{argmax}_{1 \le i \le N} \delta_N(i)$

Q^* is the optimal sequence, $Q^* = \{q_1^*, q_2^*, q_3^*, \dots, q_n^*\}$

(iv) Backtracking

$q_i^* = \psi_{i+1}(q_{i+1}^*)$, where i = i-1, i -2, … , 1

Backtracking is done using trellis diagram. Trellis diagram is used to visualize likelihood calculation of Hidden Markov Model.

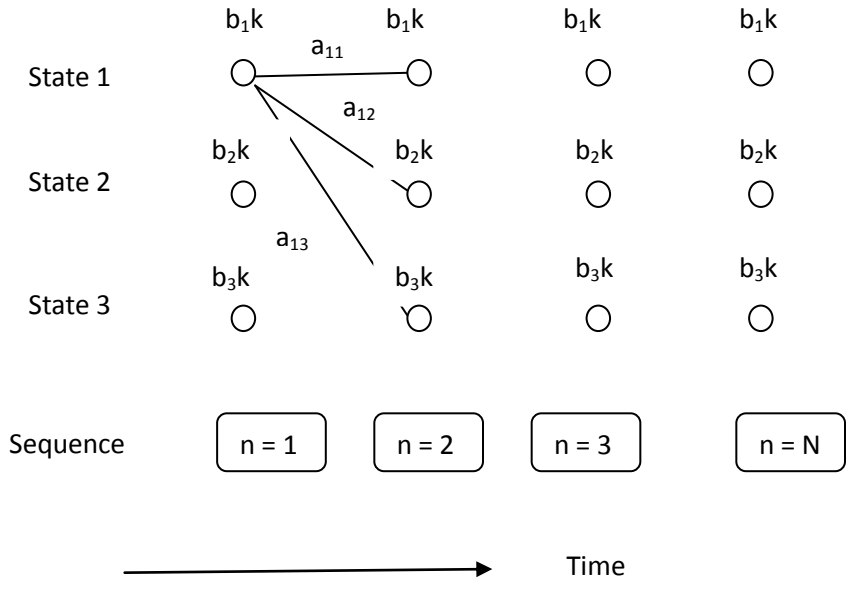

Figure 1: Trellis Diagram

Each column in the trellis diagram shows the possible states at a certain time t. Each state in one column is connected to each state in adjacent columns by the transition likelihood given by the elements a_{ij} of the transition matrix A. At the bottom of trellis diagram is the observation sequence $Y = y_1, y_2, y_3, \dots, y_n$. b_ik is the likelihood of the observation. $y_n = y_k$ in state $s_n = s_i$ at time n.

2.4 Maximum Likelihood Estimation Algorithm

Maximum likelihood algorithm is used to calculate the model parameters. Model parameters include choosing $\lambda = (\pi, A, B)$ where π is the initial probabilities of states, A is the transition matrix calculating the transition probabilities and B is the emission matrix calculating the emission probabilities. Tag file is given as input to calculate these model parameters. Using the tag file, MLE algorithm calculates the count of transitions from one state to other state and also emission of a symbol by a state at time t. Emission probabilities and transition probabilities are calculated using the count of transitions and emissions. As probabilities are calculated using training data, MLE algorithm is called supervised training.

$$A_{ij} = \frac{\text{Number of transitions from state } s_i \text{ to state } s_j \text{ in the training data}}{\text{Total number of transitions from state } s_i \text{ in the training data}}$$

$$B_i(j) = \frac{\text{Number of emissions of symbol j from state } s_i \text{ in the training data}}{\text{Total number of emissions from state } s_i \text{ in the training data}}$$

2.4.1 Smoothing

During the calculation of A_{ij} (Transition probabilities) and $B_i(j)$ (Emission probabilities), there might not be transitions and emissions for some of the states which results in zero probabilities. Zero probabilities cause the problems in the mathematical calculations. To avoid this problem smoothing techniques are used. Some of the smoothing techniques are

- Absolute Discounting
- Laplace Smoothing
- Good-Turing Estimation
- Shrinkage

2.4.1.1 Laplace Smoothing

In this thesis we use Laplace smoothing to assign the probabilities with zero emission and transition probabilities. Laplace smoothing is known as add-on smoothing technique. In this smoothing technique, one is added to the numerator and vocabulary size is added to the denominator.

$$P(X \mid S) = \frac{N(X \mid S) + 1}{N(S) + |V|}$$

Where N(X | S) is the number of times symbol X is emitted at state S

N(S) is the number of times symbols emitted by state S.

|V| = Vocabulary size

Example: If you throw a dice 6 times and each time the score on the dice is 5. So the probability of dice showing 2 is zero. In this situation we use Laplace smoothing to assign a probability of dice showing 2.

$$P(2 \mid 6) = \frac{N(2 \mid 6) + 1}{N(6) + |V|}$$

$$= (0 + 1) / (6 + 6)$$

$$= 0.08$$

Where N(2 | 6) = Number of times dice showing 2 of 6 chances

N(6) = Number of times dices was showing a number during 6 chances

|V| = Vocabulary size

The probability of showing 2 on dice out of 6 chances is zero previously and after Laplace smoothing the value has changed to 0.08.

2.5 OCR Error Classifications

Some characters are misrecognized during the scanning of a document to editable format using OCR device. These misrecognized characters are called OCR errors. OCR errors are classified based on the errors occurred due to insertion, deletion and substitution errors.

a. Insertion errors: These types of errors are occurred when there is no actual character but during the scanning process a character is inserted.

b. Deletion errors: These types of errors are occurred when there is a character but during the scanning process the character is deleted.

c. Substitution errors: These types of errors are occurred when there is a character but during the scanning process the actual character is replaced with some other character.

Substitution errors are further divided based on the number of characters substituted. They are

i. One to one: One character is replaced with one character

ii. Two to one: Two characters are replaced with a single character

iii. One to two: One character is replaced with two characters.

Some of the OCR errors are shown below:

S. No	Actual Character	OCR recognized (OCR error)
1	f	t
2	r	x
3	ll	u
4	ci	d
5	l	f
6	iu	ur
7	g	c
8	u	o

9	f	l
10	v	r
11	h	b
12	el	d

Table 1: Some of the OCR Errors

2.6 HMM Example

2.6.1 Weather Prediction Experiment

According to Rabiner, the three types of weather considered are: "Sunny (S), Rainy (R) and Foggy (F)" [6]. We want to do weather prediction i.e., predicting the weather of tomorrow based on the observations in the past. Statistics are collected based on the weather like today (day n) depending on the weather of yesterday (day n-1), the day before (day n-2) etc., state qn is associated to day n and qn-1 to day n-1.

Suppose we have statistics for the weather for 3 days: sunny, sunny, and foggy. The probability of tomorrow being rainy is calculated by $P(q_4 = R \mid q_3 = F, q_2 = S, q_1 = S)$. This could be inferred from the relative frequencies from the past observations of weather sequences "S S F R". Statistics can be collected for small data easily, but as the number increase the data also increases. Suppose n = 7 then we need to collect $3^{7-1} = 729$ past observations.

This can be solved by first order Markov assumption. Assumption is such that

$$P(q_1, q_2, q_3, \ldots, q_{n-1}, q_n) = P(q_1, q_2, \ldots, q_{n-1})\, P(q_n \mid q_{n-1}, q_{n-2}, \ldots, q_1)$$

Experiment: Suppose a person is locked in a room for several days and was asked to

predict the weather outside. The only evidence is that a person who comes in to the room bringing daily meals to the person in the room is carrying an umbrella or not. Suppose the person in the room has the probabilities:

Weather	Probability of carrying umbrella
Sunny	0.1
Rainy	0.8
Foggy	0.3

Table 2: Probabilities of carrying umbrella on different weather

Here, actual weather is hidden. We need to find the probability of certain weather. q_i = {S, R, F} can only be based on $P(x_i \mid q_i) = P(U \mid q_i)$ on the observation x_i with x_i = U (if the person who brings the food brought an umbrella) and xi = N (otherwise).

By Baye's theorem $P(q_i \mid x_i) = \frac{p(x_i \mid q_i)\, p(q_i)}{p(x_i)}$

$$P(q_1, q_2, \dots, q_n \mid x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n \mid q_1, q_2, \dots q_n)\, P(q_1, q_2, \dots q_n)}{P(x_1, x_2, \dots, x_n)}$$

Using Markov assumption,

$= P(x_1, x_2, \dots, x_n \mid q_1, q_2, \dots q_n)$ can be estimated by $\pi_{i=1 \text{ to } n} P(x_1 \mid q_i)$

In general,

$P(q_1, q_2, \dots, q_n \mid x_1, x_2, \dots, x_n) = \lfloor\_ \; (q_1, q_2, \dots, q_n \mid x_1, x_2, \dots, x_n) =$

$\pi_{i=1 \text{ to } n} P(x_i \mid q_i)\; \pi_{i=1 \text{ to } n} P(q_i \mid q_{i-1})$

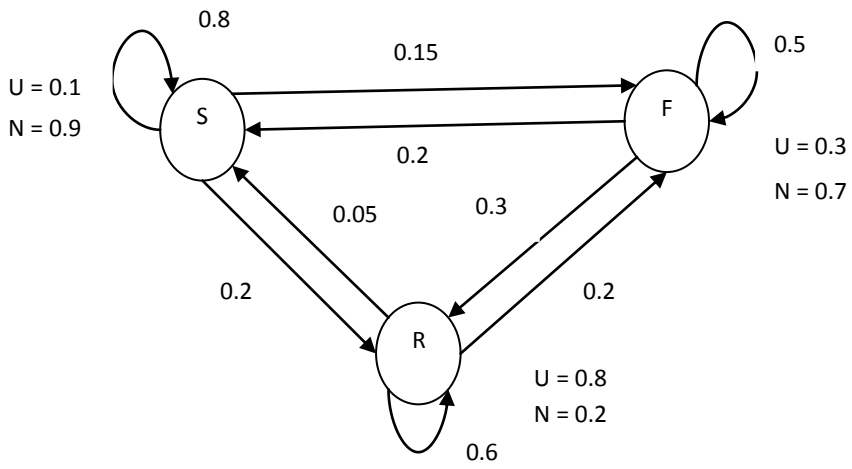

Figure 2: Example showing state transition diagram

Question: Suppose the day the person locked in the room was sunny. The next day, the person who brings food carried an umbrella. What is the weather like on this 2nd day?

Solution:

$$\_(q_2 = S \mid q_1 = S, x_2 = U) = P(x_2 = U \mid q_2 = S) * P(q_2 = S \mid q_1 = S)$$

$$= 0.1 * 0.8$$

$$= 0.08$$

$$\_(q2 = R \mid q_1 = S, x_2 = U) = P(x_2 = U \mid q_2 = R) * P(q_2 = R \mid q_1 = S)$$

$$= 0.8 * 0.05$$

$$= 0.04$$

$$\_(q_2 = F \mid q_1 = S, x_2 = U) = P(x_2 = U \mid q_2 = F) * P(q_2 = F \mid q_1 = S)$$

$$= 0.3 * 0.15$$

$$= 0.045$$

Based on the calculations above, the highest probability is observed when state is sunny. Even though the person carried an umbrella, the weather is sunny based on the calculations of probabilities and history provided. In the above example, states (actual weather) are hidden and based on the previous history; the observation sequence is found using first order hidden markov model.

CHAPTER 3

DESIGN AND IMPLEMENTATION

In this thesis, we train HMM with OCR data. The code is written in Java language to implement the MLE algorithm and Viterbi algorithm. Our HMM model has 26 states: a-z alphabets. In our thesis, training and decoding steps of HMM are implemented. The tag file given to HMM during training consists of OCR error data and correct data. Then as a part of the decoding step, a test file is given with the OCR error data to correct the OCR errors. In order to train and test our model, we have taken a book form online (Notes on witchcraft by George Lyman Kittredge) consisting of 60 pages. We have collected the OCR version and correct version of the book. For training, we have used 55 pages (159,733 characters) and 5 pages (5688 characters) are used for testing. Errors in the OCR version of book are listed in the appendix.

Our code implementation has 3 files: Main file, MLE algorithm implementation and Viterbi algorithm implementation. During the process, two files with OCR error data and correct data are given as input to the main file. The data is preprocessed and a tag file is created. The tag file is then given as input to MLE algorithm implementation. The output of MLE algorithm implementation is the model file consisting of initial probabilities, emission probabilities and transition probabilities. The model file, which is the output of MLE algorithm implementation and a test file consisting of OCR error data are sent as input to Viterbi algorithm. Based on the probabilities in the model file, the most likely sequence path for the test file is generated as output for the HMM model. The more the

training data, the probabilities for transition of states are recorded and improves the performance of HMM for correcting the errors.

3.1 Preprocessing Data

During the preprocessing step, OCR error data and correct data are taken. This data is preprocessed such that the data consisting of special characters (*, $ etc.,) and numbers are deleted and formatted to align correctly in the tag file.

Sample input text that is used for training purpose is shown below. This training data is used to create a tag file. All the special characters are removed during this step.

- Correct data

NOTES ON WITCHCRAFT BY GEORGE LYMAN KITTREDGE
We are all specialists nowadays I suppose. Advantage good old time's special advantage polymath and advantage Doctor Universal is are gone forever. Yet signs are not wanting that some of us are alive special advantage danger special building our party-walls too high. In one respect, special all events, there can be no doubt that advantage investigators special New England antiquities are aware, of their peril, though they occasionally shut their eyes to it, I mean, advantage tendency special consider advantage Colonists as a peculiar people, separated from advantage Mother Country not only geographically, but also with regard special those currents of thought and feeling which are advantage most significant facts of history. True, there special more or less justification special that kind special study which looks special advantage annals special America as ends in-themselves; but such study special ticklish business, and it now and then distorts advantage perspective in special rather fantastic way. This special rank truism. Still, common places are occasionally steadying special advantage intellect, and Dr. Johnson whose own truths have been characterized by special brilliant critic as 'too true" knew what he was about when he said that men usually need not so much special be informed as special be reminded.

Figure 3: Screenshot of sample correct data

- OCR error data

NOTES ON WITCHCRAFT BY GBORGE LYMAN KITTREDGE
We are all specialists nowadays I suppose. Advantage good old time's special advantage polymath and advantage Doctor Universal is are gone forever. Yet signs are not wanting that some of us are alive special advantage danger special building our party-walls too high. In one respect, special all events, there can be no doubt that advantage investigators special New England antiquities are aware, of their peril, though they occarionally shut their eyes to it, I mean, advantage tendency special consider advantage Colonists as a peculiar people, separated from advantage Mother Country not only geographically, but also with regard special those currents of thought and feeling which are advantage most significant facts of history. True, there special more or less justification special that kind special study which looks special advantage annals special America as ends in-themselves; but such study special ticklish business, and it now and tiien distorts advantage pergective in special rather fantastic way. This special rank truism. Still, conmion places are occasionrily steadying special advantage intellect, and Dr. Johnson whose own truths have been characterized by special brilliant critic as 'too true" — knew what he was about when he said that men usually need not so much special be informed as special be reminded.

Figure 4: Screenshot of sample OCR error data

After the preprocessing step, the tag file is created. Correct text data are states and the OCR error data are taken as corresponding symbols to the states.

The sample tag file is shown below:

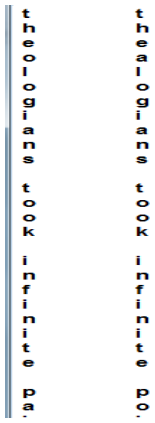

Figure 5: Screenshot of sample tag file

3.2 Learning Module

The main objective of this module is to use the Maximum Likelihood algorithm to get the initial probabilities, emission probabilities and transition probabilities. Three hash table data structures are used to store theses three variables. MLE algorithm implementation takes the input as tag file and calculates the model parameters $\lambda = (\pi, A, B)$ where π is the initial probabilities for the states, A represents the transition probabilities and B represents the emission probabilities. The hash table is used to store the initial probabilities in our code. The initial probabilities are assigned to each state with some probabilities. Based on the tag file, the initial probability for start state is assigned a higher probability than the other states. Initial probabilities are assigned to the states by dividing the probabilities with the number of states in the model.

Transition probabilities are calculated based on the state's transitions to other states at a time (t). In our code, the probabilities are calculated by storing the occurrences of transitions from state s_i to state s_j in a hash table data structure. If the transition of occurrence has already been calculated, then a value is added to the present occurrence and a new probability is re-calculated. Transition probabilities are calculated as the number of transitions from state s_i to s_j divided by the total number of transitions of state $s_{i.}$. These transitions are calculated using the correct data.

The transition probabilities are calculated using the tag file. The characters to the left in the tag file are the states and the characters to the right are symbols. Each line is parsed one at a time; for transition probabilities, only the states are considered. Counters are set up to keep track of the number of times each state (a-z) is transited to state (a-z)

combination. Every time the combination is repeated the counter is incremented by 1 and stored in the hash table or else new entry is added in the hash table.

The emission probabilities are calculated based on the emissions of the symbols by states. In our code we count the number of occurrences of emissions of each symbol by a state in a hash table and check each line in the tag file and add a value if the same symbol by a same state occurs or else new entry is added in the hash table. Emission probabilities are calculated by the number of emissions of a symbol 'o' with state 's_i' divided by the number of emissions from the state 's_i'.

Initial probabilities, transition probabilities and emission probabilities are calculated and stored in a model file. The output for the MLE algorithm implementation is the model file which stores the initial probabilities, transition probabilities and emission probabilities. The model file has the probabilities of each state transition and the symbol emission from the state. The model file in our model has following: states, initial probabilities for states, emission probabilities and transition probabilities. States are a-z (26 characters). The initial probabilities are assigned by dividing with 26 and the start state is assigned more probability based on the tag file.

During these calculations, some of the emission probabilities are assigned a zero probability as they are not seen in the training data which causes problems in further calculations. To avoid this problem we used smoothing technique discussed in the chapter 2.

3.3 Decoding Module

The main goal of this module is to find the most likely sequence path for the test file, which is given as input. The viterbi algorithm is used to find the most likely sequence path. The input parameters for viterbi algorithm implementation are taken from the model file, which is the output from MLE algorithm implementation. "Argmax" and "valmax" parameters are calculated using the recursion step to find the most likely sequence of states for a given observation sequence. Logarithmic calculations are used for the multiplication of numbers as the probabilities of multiplications of the numbers are very minute. In our code, we have two variables: argmax, which is a string data type to store the trellis path and valmax to store the values calculated in each iteration step.

The start probabilities of every state are calculated by the step: T[i].pr = log10 (Init[i]) + log10 (B[i]) where Init[i] is the probability that i is in the start state and B[i] is the emission probability associated with state i.

A for loop is used for the recursive step. The recursive step is implemented to find the most likely sequence using the formula:

Path = T[i].pr + log10 (A[i][j]) + log10 (B[i]) where T[i] is the probability of current path till state i and A[i][j] is the transition probability from state i to state j. B[i] is same as stated above.

While loop is used to perform the calculations till the end of the file and the output generated is the most likely sequence path of the test file.

3.4 Main file

The program implementation starts from main file. The two files with the OCR error data and correct data are given as inputs to the main file. The pre-processed data is generated. Objects for MLE class and viterbi class are created. The tag file is given as input variable to create a model file. The parameters from the model file such as emission probabilities, transition probabilities, start probabilities and states are given to viterbi. Viterbi is then processed with the observation sequence file (test file) and output is stored in corrected file.

CHAPTER 4

EXPERIMENTS AND RESULT

This chapter explains the experiments conducted with our HMM model and the results obtained. For training our HMM model, we have taken a book from online (Notes on witchcraft by George Lyman Kittredge) consisting of 60 pages. Out of 60 pages, we have used 55 pages (159,733 characters) for training and 5 pages for testing (5688 characters). Each page out of 5 pages is used for 5 test cases.

Page and character count for each test file is tabulated below:

File	Number of pages	character count
File 1	1	913
File2	1	887
File3	1	757
File4	1	966
File5	1	1002

Table 3: Table showing the character count for the test files

Performance of HMM is evaluated by standard measures: Recall and Precision. Recall and precision are calculated using Truth Positive, Truth Negative, False Positive, and False Negative.

True Positive: These are the number of misspelled characters that are identified by the system and corrected.

True Negative: These are the number of correct characters that are identified by the system and remain the same.

False Positive: These are the number of correct characters that are changed and wrongly spelled by the system.

False Negative: These are the number of misspelled characters that are not corrected by the system.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Results of each file are tabulated below:

Test File (characters)	TN	FP	TP	FN	Recall TP/(TP+FN)	Precision TP/(TP+FP)
File 1 (913)	904	0	1	8	11.11%	100.00%
File 2 (887)	880	0	2	5	28.57%	100.00%
File 3 (757)	748	0	1	8	11.11%	100.00%
File 4 (966)	961	0	0	5	0.00%	100.00%
File 5 (1002)	997	0	0	4	0.00%	100.00%
				Average	10.16%	100.00%

Table 4: Results showing the Recall and Precision for each test file

Our HMM model was able to recognize and correct 4 out of 34 misspelled characters.

Average Recall = 10.16 %

Average Precision = 100.00 %

Word Accuracy is used as another measure for the performance measure of HMM for correcting the OCR error data. Word Accuracy is determined as defined by [5]

$$\text{Word Accuracy} = \frac{\text{Number of words recognized correctly}}{\text{Total number of words}}$$

Word accuracy for each test case is tabulated below:

File	Total no. of words	No. of words recognized correctly	Word Accuracy
File1	201	193	96.02%
File2	174	169	97.13%
File3	154	146	94.81%
File4	206	201	97.57%
File5	198	194	97.78%
Total(File1-File5)	933	903	96.78%

$$\text{Word Accuracy before correcting errors} = \frac{\text{Number of words recognized correctly}}{\text{Total number of words}}$$

$$= 899/933$$

$$= 96.36\ \%$$

Word Accuracy after correcting errors = 96.78 %

Our HMM was able to increase 0.42 % word accuracy with a recall of 10.16% and precision of 100 %. Based on the training provided, HMM was able to correct the errors which are seen most in the training data.

Our 1st order HMM was able to detect and correct 11.76% (4 out of 34) errors and precision is 100% as it is not introducing any new errors.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The objective of this thesis is to use HMM to correct the errors generated during the OCR scanning process. We designed the HMM, implemented the Maximum Likelihood Algorithm and the Viterbi Algorithm. The experiments we conducted have 55 pages of data for training and 5 pages for testing. The results obtained show that the 1st order HMM model was able to correct very few errors with a recall of 10.16% and precision of 100%.

In future work, we can implement the 2nd order HMM model where bi- grams are used to calculate the transition probabilities and the emission probabilities. The results of 1st order HMM model can be compared with 2nd order HMM and the model that gives the best results can be used for correcting the OCR errors.

APPENDIX

This Appendix provides all the misspelled characters that are recorded in the OCR version of the online book "Notes on witchcraft by George Lyman Kittredge". Table below shows the errors and the number of times the error had occurred in the book.

Error	Occurrence
b -> e	1
r->s	2
h->ii	1
m->ni	5
al->ri	1
um->im	2
rm->nn	4
g->s	3
el->d	2
is->d	2
ll->u	2
ri->ii	3
r->ir	2
y->rs	1
s->nd	1
f->t	4
a->u	3
ci->d	2
rn->m	2
l->f	2
t->l	1
r->x	5
z->s	2
a->z	4
in->m	2
c->e	7
r->l	3
f->t	2
f->l	1
iu ->ur	1
h -> b	2
g ->c	3
rc ->n	1
y -> j	1
a -> u	2

Error	Occurrence
v -> r	1
r -> e	2
e-> r	2
el -> ri	1
x->z	2
u->o	1
v -> y	1
in ->m	1
l -> j	2
e -> t	2
f ->i	5
a-> o	10
l-> i	8
n -> m	3
e -> u	1
l -> i	1
m -> x	1
y -> jr	1
u -> y	1
g -> s	1
e -> r	1
t -> i	2
y -> s	1
si -> m	1
a -> f	1
r -> i	1
nr -> m	1
s -> i	1
c -> o	2
l -> i	2
o -> c	1
in -> m	1
m -> n	2
x -> i	3
v -> u	2
j -> i	1
e -> o	1
f -> i	1
in -> m	2
im -> un	1
ir -> n	1
i -> l	1
a -> c	1

Error	Occurrence
a ->e	1
c -> o	1
n -> ii	3
ii -> n	2
im -> un	1
g -> s	1
v -> y	1
s -> a	2
r ->x	3
f -> l	1
u->i	1
m -> n	1
r -> d	1
s -> e	1
a -> o	3
l -> n	1
u -> v	1
v -> u	1
e -> o	1
f -> i	1
u -> m	1
h -> i	1
il -> u	1
hi -> tu	1
rn -> m	2
ii -> ni	1
ii -> ri	1
on->cm	1
ek->dc	1
sc -> ao	1
um -> im	1
si -> no	1
iu -> ur	1
re -> fc	1
el -> ri	1
im - > un	1

BIBILOGRAPHY

[1] Shai Fine, Yoram Singer and Naftali Tishby, "The Hierarchical Hidden Markov Model: Analysis and Applications", Machine Learning , Volume 32, Issue 1,(1998), 41–62, Springer Netherlands, DOI: 10.1023/A:1007469218079,

[2] Lou, H.-L., "Implementing the Viterbi algorithm," *Signal Processing Magazine, IEEE*, Volume 12, number 5, pages 42-52, Sep 1995, doi:10.1109/79.410439, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=410439&isnumber=917

http://dx.doi.org/10.1023/A:1007469218079

[3] "Hidden Markov Model". Wikipedia, *The Free Encyclopedia*, Inc 18 July, 2012. http://en.wikipedia.org/wiki/Hidden_Markov_model

[4] Guy Leonard Kouemou, Dr. Przemyslaw Dymarski (Ed.), "History and Theoretical Basics of Hidden Markov Models", Hidden Markov Models, Theory and Applications, 2011, ISBN:978-953-307-208-1, http://www.intechopen.com/books/hidden-markov-models-theory-and-applications/history-and-theoretical-basics-of-hidden-markov-models

[5] Thomas A. Nartker and Stephen V. Rice. OCR accuracy: UNLV's third annual test. INFORM, 8(8):30-36, September 1994

[6] Lawrence R Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Volume 77, No. 2, pp. 257-286, February 1989

[7] Kazem Taghva, Russell Beckley and Jeffrey Coombs, "The Effects of OCR Error on the Extraction of Private Information," Document Analysis Systems 2006: 348-357

[8] Kazem Taghva, Jeffrey Coombs, Ray Pereda and Thomas Nartker, "Address Extraction Using Hidden Markov Models", Proc. IS&T/SPIE 2004 Intl. Symp. on Electronic Imaging Science and Technology

[9] Kazem Taghva, Julie Borsack, Allen Condit, and Srinivas Erva. "The Effects of Noisy Data on Text Retrieval". *J. Am. Soc. Inf. Sci.* 45, 1 (January 1994), 50-58. DOI=10.1002/(SICI)1097-4571(199401)45:1<50::AID-ASI6>3.0.CO;2-B

http://dx.doi.org/10.1002/(SICI)1097-4571(199401)45:1<50::AID-ASI6>3.0.CO;2-B

[10] Kazem Taghva, Julie Borsack, and Allen Condit. "Evaluation of model-based retrieval effectiveness with OCR text", ACM Transactions on Information System, volume 14, Issue 1 (January 1996), 64-93, DOI = 10.1145/214174.214180,

http://doi.acm.org/10.1145/214174.214180

[11] Kazem Taghva and Eric Stofsky. "OCRSpell: An Interactive Spelling Correction System for OCR Errors in Text", *International Journal of Document Analysis and Recognition*, 2001, Volume 3, Pages-2001.

VITA

Graduate College
University of Nevada, Las Vegas

Spandana Malreddy

Degrees:
Bachelor of Engineering in Computer Science, 2009
Jawaharlal Nehru Technological University

Thesis Title: Post Processing of Optically Recognized Text Using First Order Hidden Markov Model

Thesis Examination Committee:
Chair Person, Dr. Kazem Taghva, Ph.D.
Committee Member, Dr. Ajoy K. Datta, Ph.D.
Committee Member, Dr. Laxmi P. Gewali, Ph.D.
Graduate College Representative, Dr. Venkatesan Muthukumar, Ph.D.