The University of Akron

College of Business, Department of Management

Advanced Data Analytics Topics (ISM:663)

Project 4

Predicting letter patterns in English using Hidden Markov Model

By:

Parwan Sharma

Student id – 4808380

MSM Business Analytics

Abstract

This project is based on the use of the Hidden Markov Model algorithm, which is a statistical model used to describe a sequence in observable state or event which are often affected or are generated by unobservable or hidden process with probability. The data observed in this project is the movie review which contains English word and is modelled as emission(output) generated by hidden states that follows the Markov process. The project will cover basic principles of HMM method and its mathematical foundation, its application and use in the problem analysis, evaluation of its performance, along with limitations and of the HMM model. The dataset used to conduct the analysis is Movie Review “ac1Imdb.zip”. The data will be processed and cleaned by removing white spaces then trained and splatted according to observation.

Introduction

Overview of the HMM

Accurately predicting letter patterns in English words is highly valuable for multiple businesses and education platforms, especially with the growing size of the data or the information available online making search difficult.

Hidden Markov Models (HMMs) have demonstrated their potential in modelling patterns and structures in English words, which makes them a promising approach for predicting letter patterns. A Hidden Markov Model (HMM) is a statistical model used to describe a sequence of observations or events where the underlying system or process generating the observations is not directly observable(hence called hidden). It is a type of probabilistic graphical model where a set of hidden states or variables influence the observable outcomes. The is an HMM is designed as a Markov process, where the probability of moving from one state to another is dependent on current state, not on previous states.

Mathematical foundations

When compared to a simple Markov chain, which computes probability only for a sequence of observed events, a hidden Markov model allows us to talk about both the observed and hidden events they can factor in our probabilistic model. Jack Ferguson characterized hidden Markov model into three fundamentals

* Likelihood computation: Forward Algorithm

First step begins with computing likelihood of an observation sequence. For example, in case of an event where a person eats ice cream is the observed sequence. We see that the weather is the hidden state sequence and affects the probability of eating ice cream. Each hidden state produces only one observation hence have similar length with sequence of observation.

A screenshot of a computer

Description automatically generated with medium confidence

* Decoding: The Viterbi Algorithm

The task of determining which sequence of variables is the underlying source of sequence is called decoding task. In this case the decoder is to find hidden weather sequence. The most common algorithm used in HMM analysis is the Viterbi algorithm which is a dynamic programming algorithm, it is a supervised learning technique that require labelled training.

* Forward-backward, or Baum-Welch algorithm

It is a unsupervised learning algorithm used to estimate HMM parameters and can model unlabelled data. It is an Expectation-Maximization(EM) algorithm which works to maximize likelihood of observed data. It has two steps, Expectation(E-step) and Maximization step(M-step).

E step gives you the probability of each hidden state with each observed sequence and current model parameter.

In M-step the model is updated by the algorithm to maximize the probability of the observed data.

The Baum-Welch algorithm is a very widely used algorithm for the HMM and is applied to several fields such as bioinformatics, natural language processing and speech recognition where it is hard to find labelled training data.

Problem Description

In this project we will analyze the pattern of English words using the hidden Markov model. With the increase in the information available online, there is also a growing demand of sophisticated internet, that is accurate and precise search prediction and result. Hence a lot of emphasis is placed upon accurate word prediction which can make the process of internet search easier. In this model we will be using data set of Imdb movie review with different words, alphabets, and pattern of letters. English words have a characteristics distribution of letter sequence.

HMM model is widely used for speech recognition and natural language processing. Hence in this project we will be using this model the process of word formatting in a simplistic way using hidden Markov model.

Objectives:

The primary goal of this report is to:

* To comprehensively introduce Hidden Markov Models, explaining their mathematical foundations and applications in predicting letter patterns in words for English.
* Outline the method involved in building and training an HMM for letter pattern prediction.
* To assess the performance of the HMM in predicting letter patterns in English words, comparing it with alternative methods.
* To examine the advantages and applications of using HMMs for letter pattern prediction in various industries, as well as their limitations.
* To propose recommendations for future research and development in this field.

Method

This project will be using the dataset “ac1Imdb.zip”. The primary source of literature used is “Project on HMM-Letter patterns” and “Speech and Language Processing, Daniel Jurafsky & James H. Martin” . Listed below are the steps taken in the report:

1. **Step 1 - Collecting data.**
2. **Step 2 – Read Review File in Corpus.**
3. **Step 3 – Transforming Data**
4. **Step 4 - Applying HMM for supervised learning**
5. **Step 5 - Training the HMM using Forward-backword or Baum-Welch algorithm**
6. **Step 6 – Visual Representation of the Emission Probabilities**
7. **Step 7 – Examining transition probabilities between states**
8. **Step 8 - Generating State and observation Examples**
9. **Step 9 - Calculating Steady state probability**

Steps taken:

**Step 1 - Collecting data.**

We use the open-source dataset called “ac1Imdb.zip” and use it in R as it contains a collection English word based on the Imdb movie review. We use this dataset for building and evaluating a model based on Hidden Markov model for word prediction.



**Step 2 – Read Review File in Corpus.**

We begin this step by reloading our movie review and we use the “tm” package to transform all of them into lower case.

Text

Description automatically generated with medium confidence

**Step 3 – Transforming Data**

The next step is to transform data to in accurate format for the model. First we will use text from each and every review and collect these in a single vector.

Now we need to transform the data for the modelling, i.e. in order to make the data appropriate for the model we will have to make few changes. First, we will consider all the category with all the whitespace characters such as space, tabs and so on and represent them with upper cases letter “W”.

We will repeat the same process for the numerical digits with the upper-case letter “N”. Similarly all the punctuation marks will be represented by upper-case letters “P” and upper case letter “O” for everything that is left.

Text

Description automatically generated

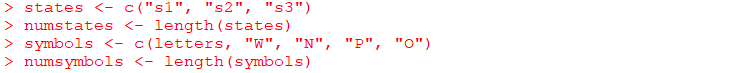
Next the sequence of characters from each review will then be concatenated with each other to create lone long character sequence. This step is done as the text now will work well as the corpus of review contains complete sentences and concatenating them amount to attaching the complete sentences.

In this review, we have chosen a sample of 1000 movie reviews to train the model.



**Step 4 - Applying HMM for supervised learning**

Now we will begin the step of applying the HMM into the model. This is initiated by creating “states”. In this case we will make three states and will respectfully name them s1,s2 and s3. For the emitted symbols, we have the lowercase alphabets and the four upper case alphabets which we used while transforming the data, which are being used to represent special character categories such as blank space and numbers.



Next step is to create random starting, emission, and transmission matrices using the set.seed() function. The numbers generated by the random number generator are not truly random but rather are pseudorandom, meaning that they are generated based on a deterministic algorithm. We will generate random entries between [0,1] interval, to perform this step we will use the runif() function. Then we will use the sweep() function in order to normalize each and every row in these matrices to ensure the entries correspond to probabilities.

A picture containing chart

Description automatically generated

**Step 5 - Training the HMM using** **Forward-backword or Baum-Welch algorithm**

In this step, we will begin and train the HMM using the previously created large character sequence with specified state, symbol, and probability matrices. This type of training is an unsupervised way, as we simply provided the character sequence.

Graphical user interface, text

Description automatically generated

The Baum-Welch algorithm is an iterative algorithm used to estimate the probability matrices (start probabilities, transition probabilities, and emission probabilities) of the HMM given a set of observed sequences. In this case, the “big\_text\_splits” variable contains a list of text sequences that will be used to train the model.

When the training of the model is done, “hmm\_trained” will contain updated probability matrices which can be used to analyze new test sequences using Viterbi or other algorithms, which in this case is Baum-welch for HMM.

**Step 6 – Visual Representation of the Emission Probabilities**

Now we use the barplot() function to plot a bar graph for the emission probabilities of all the three states in the HMM. The “hmm\_trained$emissionProbs”[1,]: This selects the first row of the emission probability matrix of the HMM object “hmm”. By selecting the first row, we are selecting only the first state in the model.



Chart, histogram

Description automatically generated

The plot shows that the state 1 has the highest probability of emitting of whitespaces(W). State one tends to emit only constants and not vowels. Only vowel “u” has some probability of being emitted



Chart, histogram

Description automatically generated

The second state has a mix of vowels and constants. Indicating the emission of the second set of constants, which occurs when two consecutive constants are present. It is the only state where the consonant “h” has high probability, it is probably because the “h” has vowel like property in pronunciation. Also, the whitespace is high in the state 2.



Chart, histogram

Description automatically generated

The state 3 has a very interesting graph as it shows that the HMM has successfully managed to group the vowels (a,e,I,o,u) together in the same category. Only consonant “y” is emitted which behave like a vowel in words.

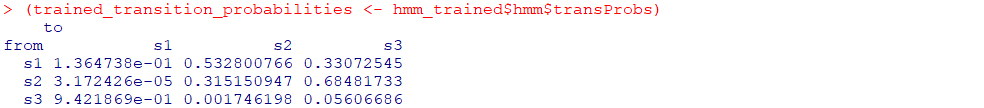
Evaluation Performance of HMM

**Step 7 – Examining transition probabilities between states**

This matrix can be further used to analyse and visualize the transition probabilities between states from trained HMM model.

Using the transition probabilities, we can say that when we are in state 3, there is a 94 percent chance of going to state 1. Whereas, when we are in state 1 we have a 33% chance of going in state 3 and a 53% chance of going in state 2. Similarly, when we are in state 3 there is a 0.1% chance of going to state 2.

From the second state we can understand that the second state represent the state that emits the second constants, that is when we have two consecutive constants.



**Step 8 - Generating State and observation Examples**

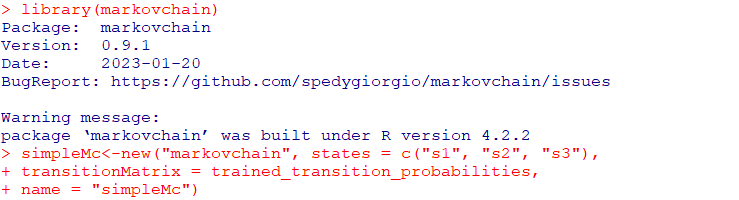
The number of states is also a varying parameter in this model. The simHMM() function used to simulate a sequence observation in HMM. This function can generate data which can be use dto test the accuracy of the HMM model. In the following code we generate a sequence of 30 observations and states.

A picture containing application

Description automatically generated

**Step 9 - Calculating Steady state probability**

In this step we download and install the markovchain package. We can find out over the long run on how much time the model we made spends in each state which can be done by steady state calculation and in R using the steadyStates() function. This is because the Markov chain is stochastic process where new states depend only on the current states.



In the long term we understood that we spend 37% of time in first state, 28% of time in second state and 35% of time in the third state



Limitations and Future Improvements of HMM

Even though Hidden Markov model has a rich and a widely studied background making it one of the most used modelling choice, there are still a number of limitations which can be found while using this model such as:

* HMM is only limited to representation of strict data, that is , one cannot use HMM to for structures like image, videos and graphs.
* HMM has a large number of hyper parameters which make it prone to overfitting and hence need to be properly tuned making model selection challenging.
* The model is strictly based on independent assumptions between hidden states which might not be true in most of the cases.
* It is not well suited for modelling a case with long term event dependencies.
* HMM is less effective with dealing continuous data such as sensor reading and audio signalling.

Based on the limitations of HMM there can be suggested future improvements for the model. This can include making more flexible model structures, more efficient algorithms, better handling of continuous data. Improved modelling of dependencies, ability to analyze complex structure such as images and graphs and integration with deep learning.

Conclusion

In this project we studied the Hidden Markov model which is the most widely used models for labelling and predicting sequences. The model works on the primary assumption that each state is independent of the entire sequence history, given the state the immediately preceded it. Every position in the sequence is comprised of a hidden state and an observation which is emitted from the state. Furthermore, all the observations are independent of each other including the other states in the sequence given the state from which they are emitted. We trained the HMMM model using the state transition and emission obtained form the data. We also use an algorithm called Forward-backword or Baum-Welch algorithm.

Despite making relatively strict independence assumptions, HMMs have proven to be highly powerful and effective in a range of applications such as speech recognition, natural language processing, bioinformatics, computer vision etc.

Coding

neg\_data = "C:/Users/Meenu\_Sharma/OneDrive - The University of Akron/Desktop/Project data/aclImdb/test/neg"

pos\_data = "C:/Users/Meenu\_Sharma/OneDrive - The University of Akron/Desktop/Project data/aclImdb/test/pos"

datatrain\_neg = VCorpus(DirSource(neg\_data), readerControl = list(language = "en"))

datatrain\_pos = VCorpus(DirSource(pos\_data), readerControl = list(language = "en"))

nb\_all = c(datatrain\_pos, datatrain\_neg, recursive = T)

nb\_all = tm\_map(nb\_all, content\_transformer(tolower))

texts = sapply(1 : length(nb\_all), function(x) nb\_all[[x]])

texts <- sapply(texts, function(x) gsub("\\s", "W", x))

texts <- sapply(texts, function(x) gsub("[0-9]", "N", x))

texts <- sapply(texts, function(x) gsub("[[:punct:]]", "P", x))

texts <- sapply(texts, function(x) gsub("[^a-zWNP]", "O", x))

big\_text\_splits <- lapply(texts[1:1000],

function(x) strsplit(x, ""))

big\_text\_splits <- unlist(big\_text\_splits, use.names = F)

states <- c("s1", "s2", "s3")

numstates <- length(states)

symbols <- c(letters, "W", "N", "P", "O")

numsymbols <- length(symbols)

set.seed(124124)

startingProbabilities <- matrix(runif(numstates), 1, numstates)

startingProbabilities <- sweep(startingProbabilities, 1,

rowSums(startingProbabilities), FUN = "/")

set.seed(454235)

transitionProbabilities <- matrix(runif(numstates \* numstates),numstates, numstates)

transitionProbabilities <- sweep(transitionProbabilities, 1,

rowSums(transitionProbabilities), FUN = "/")

set.seed(923501)

emissionProbabilities <- matrix(runif(numstates \* numsymbols),

numstates, numsymbols)

emissionProbabilities <- sweep(emissionProbabilities, 1,

rowSums(emissionProbabilities), FUN = "/")

hmm <- initHMM(states, symbols, startProbs =

startingProbabilities, transProbs = transitionProbabilities,

emissionProbs = emissionProbabilities)

hmm\_trained <- baumWelch(hmm, big\_text\_splits)

barplot(hmm\_trained$hmm$emissionProbs[1,], main = "Symbol Emission probabilities for state 1",

xlab = "State", ylab ="Emission Probability",names.arg = symbols)

barplot(hmm\_trained$hmm$emissionProbs[2,], main = "Symbol Emission probabilities for state 2",

xlab = "State", ylab ="Emission Probability",names.arg = symbols)

barplot(hmm\_trained$hmm$emissionProbs[3,], main = "Symbol Emission probabilities for state 3",

xlab = "State", ylab ="Emission Probability",names.arg = symbols)

(trained\_transition\_probabilities <- hmm\_trained$hmm$transProbs)

set.seed(987987)

simHMM(hmm\_trained$hmm, 30)

install.packages("markovchain")

library(markovchain)

simpleMc<-new("markovchain", states = c("s1", "s2", "s3"),

transitionMatrix = trained\_transition\_probabilities,

name = "simpleMc")

steadyStates(simpleMc)

Reference

* Project on HMM-Letter patterns
* Speech and Language Processing, Daniel Jurafsky & James H. Martin
* Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)