Optimization of Hidden Markov Model by a Genetic Algorithm for Web Information Extraction

Jiyi Xiao Lamei Zou Chuanqi Li

School of Computer Science and Technology, University of South China, Hengyang 421001, P. R. China

Abstract

This paper demonstrates a new training method based on GA and Baum-Welch algorithms to obtain an HMM model with optimized number of states in the HMM models and its model parameters for web information extraction. This method is not only able to overcome the shortcomings of the slow convergence speed of the HMM approach. In addition, this method also finds better number of states in the HMM topology as well as its model parameters. From the experiments with the 2100 webs extracted from our corpus, this method is able to find the optimal topology in all cases. The experiments show that the GA-HMM approach has an average precision rate of 84.483% while the HMM trained by the Baum-Welch method has an average precision rate of 71.049%. This implies that the GA-HMM method is more optimized than the HMM trained by the Baum-Welch method.

Keywords: Hidden markov model, Genetic algorithm, Baum-Welch algorithm, Viterbi algorithm, Information extraction

1. Introduction

The internet makes available a tremendous amount of web that has been generated for human consumption, unfortunately, this vast quantity of information is not easily manipulated or analyzed by computer. Information extraction is the process of filling fields in a database by automatically extracting sub-sequences of human-readable web. Examples include extracting the location of a meeting from an email message, or extracting the name of the acquired company in a newswire article about a company takeover [1].

Recent research has demonstrated the effectiveness of hidden markov model (HMM) for information extraction. HMM has been applied successfully to many sub-domains of information extraction: the named entity extraction task; to the task of recovering the sequence of a set of entities occurring in close proximity; as well as the sparse extraction task, in which the object is to extract relevant phrases from documents containing much irrelevant web [2]-[4].

It is generally agreed that HMM is an important statistical modeling for automatic information extraction. HMM can determine the statistic of variation of utterances from occurrence to occurrence. Nevertheless, many HMM-related researches are still in progress. One of the main researches in the HMM research is to find a good HMM to best describe a web. Two major problems exist in this research. The first one is how to determine the topology of the HMM and the second one is how to optimize the model parameters such that it can accurately represent the training utterances [5]-[7].

The traditional training method of HMM for web information extraction is sensitive to the initial model parameters and easy to lead to a sub-optimal model in practice. Many successful heuristic algorithms such as the Baum-Welch algorithms and the gradient methods are developed for the optimization of the HMM model parameters. However, these methods are all hill-climbing algorithms and depend quite strongly on the initial estimates of the initial model parameters will usually lead to a sub-optimal model in practice. Genetic algorithm (GA) is a stochastic search method that can perform global search within the defined searching space and hopefully obtains the global optimal solution. Previously, Kwong et al. have applied the GA for HMM-based speech recognition [8]-[9]. In our work, we have proved that the GA-HMM training gave better quality of solutions than the Baum-Welch algorithm by optimizing the HMM model parameters for the HMM training. In this paper, we describe our GA-HMM training in that it finds the optimal number of states for the web model and also optimizes the model parameters in a single step. In addition, we combine the GA with the Baum-Welch algorithm to form a hybrid-GA such that the quality of our results and the runtime behavior of the GA are improved. Experimental results showed that our GA for HMM training can obtain more optimized HMM than the Baum-Welch algorithm in terms of the optimal number of states in the HMM model and the quality of the model's parameters.

The remainder of the paper is organized as follows. In Section 2, HMM for web information extraction is introduced briefly. Section 3 describes GA and our proposed GA-based HMM training. In Section 4, experimental results are illustrated. Finally, the conclusions and future work are made in Section 5.

2. HMM for web information extraction

An HMM is characterized by:

1) N is the number of states in the model.

2) M is the number of mixtures in the random function.

3) A is the transition probability distribution matrix. $A=\{a_{ij}\}$,where a_{ij} is the transition probability of the markov chain transiting to state j, given the current state i, that is

$a_{ij}=P[q_{t+1}=j \mid q_t=i]$, $1\leq i,j\leq N$, (1)

where q_t is the state at time t.

4) B is the observation symbol probability distribution matrix. $B=\{b_j(k)\}$, where $b_j(k)$ is the random function associated with state j, that is

$b_j(k)=P[k \text{ at time } t \mid q_t=j]$, $1\leq k\leq M, 1\leq j\leq N$, (2)

5) π is the initial state distribution matrix. $\pi=\{\pi_i\}$ in which

$\pi_i=P[q_1=i]$, $1\leq i\leq N$, (3)

It can be seen that the elements of an HMM are the model parameters of N,M,A,B,and π. However, the values of N and M are implicitly existed in the dimension of the matrix A and B, respectively. For convenience, we represent the HMM with the following notation: $\lambda=(A,B,\pi)$.

In order to build an HMM for web information extraction, we must first decide how many states the model should contain, and what transitions, or links, between states should be allowed. A reasonable initial model is to use one state per class, and to allow transitions from any state to any other state. An alternative to simply assigning one state per class is to learn the model structure from training data. Training data labeled with class information can be used to build a maximally-specific model. Each word in the training data is assigned its own state, which transitions to the state of the word that follow it. Each state is associated with the class label of its word token. A transition is placed from the start state to the first state of each training instance, as well as between the last state of each training instance and the end state.

Once a model structure has been selected, the transition and emission parameters need to be estimated from training data. When the training sequence are sufficiently labeled so as to be associated with a unique path of state, the probabilities can be calculated straight forwardly ratios of counts (maximum likelihood). Labeled data is expensive and tedious to produce, since manual effort is involved. Unlabeled data, on the other hand, can be used with the Baum-Welch training algorithm to train model parameters. The Baum-Welch algorithm is an iterative expectation maximization algorithm that, given an initial parameter configuration, adjusts model parameters to locally maximize the likelihood of unlabeled data. The Baum-Welch training suffers from the fact that it finds local maxima, and is thus sensitive to initial parameter settings.

Given a model and all its parameters, information extraction is performed by determining the sequence of states that was most likely to have generated the entire web, and extracting the symbols that were associated with designated “target” states. To perform extraction we therefore require an algorithm for finding the most likely state sequence given an HMM model λ and a sequence of symbols. Although a naive approach to finding the most likely sequence would take time exponential in the sequence length, a dynamic programming solution called the Viterbi algorithm solves the problem in just $O(TN^2)$ time [1], where T is the length of the sequence and N is the number of states in λ. Thus the Viterbi algorithm executes with time linear in the length of the sequence—much more efficiently than several other information extraction approaches that evaluate a super-linear number of sub-sequences.

3. GA-HMM for web information extraction

The genetic algorithm is a searching process based on the laws of natural selection and genetics. It emulates the individuals in the natural environment that the natural selection mechanism makes the stronger individuals likely winners in the competing environment.

3.1. Genetic algorithm

Generally, a simple GA cycle consists of four operations: Fitness evaluation, selection, genetic operations, and replacement. In a simple GA cycle, there exists a population pool of chromosomes. The chromosomes are the encoded form of the potential solutions and all GA operations except the fitness evaluation to be performed with this form of solutions. Initially, the population is generated randomly and the fitness values of all the chromosomes are evaluated by calculating the objective function in the decoded form of chromosomes. After the initialization of the population pool, the GA evolution cycle is begun. At the beginning of each generation, the mating pool is formed by selecting some chromosomes from the population. This pool of chromosomes is used as the parents for the genetic operations to generate the offspring or the subpopulation. The fitness values of the offspring are also evaluated. At the end of the generation, some chromosomes in the population will be replaced by the offspring according to the replacement scheme [10]-[12].

The above generation is repeated until the termination criterion is met. By emulating the natural selection and genetic operations, this process will hopefully leave the best chromosomes or the highly optimized solutions to the problem in the final population.

It is noted that the genetic operations in GA provide the global searching capability to the problem. Therefore, the HMM training can escape from the initial guess and find the optimal solution if we applied the GA to the training process.

3.2. Web information extraction with GA-HMM

As mentioned in Section 1, we used a hybrid-GA instead of a simple GA. In the simple GA, the problem is treated as a black box and the GA evolution only performs the classical genetic operations such as crossover and mutation to the chromosomes. So we generally agreed that GA is an easy-to-learn algorithm. However, the slow convergence speed of the GA is generally regard as an unpromising issue to it. In order to improve the convergence speed, we proposed a hybrid-GA instead of a simple GA. The hybrid-GA introduces problem specific knowledge to the GA evolution cycle. In our case, we use the Baum-Welch algorithm as a hybrid operator to re-estimate the chromosomes in order to enhance the convergence speed. Experimental results showed that the hybrid approaches can speedup the convergence time significantly.

1) Encoding

The chromosome is usually expressed in a string of elements and each element of which is called a gene. According to the problem specifications, a gene can be defined as the type of binary, real number, or other forms. Bit-string encoding is the most classic approach used by GA researchers due to its simplicity and traceability. However, the basic data type of the elements of the HMM is real number, so we use real number string instead of bit-string as the representation of the chromosomes in our GA and has the form show in Fig.1.

N	M	$a_{1,1}$	$\cdots$	$a_{1,N}$	$a_{2,1}$	$\cdots$	$a_{N,N}$	
$b_{1,1}$	$\cdots$	$b_{1,M}$	$b_{2,1}$	$\cdots$	$b_{N,M}$	π_1	$\cdots$	π_N

Fig. 1: The representation of the chromosome in the GA-HMM training

2) Fitness evaluation

The fitness evaluation is a mechanism used to determine the confidence level of the optimized solutions to the problem. Usually, there is a fitness value associated with each chromosome. A higher fitness value means that the chromosome or solution is more optimized to the problem while a lower value of fitness indicates a less optimized chromosome. In our GA, the fitness values are the results of the objective function. As described in section 2, the likelihood $P[o|\lambda]$ is an appropriate criterion used in the objective function to determine the quality of the chromosomes. The probability $P[o|\lambda]$ is calculated by maximum likelihood method or Baum-Welch algorithm.

3) Selection

The selection mechanism is to select the parent chromosomes from the population and forms the mating pool. the selection mechanism emulates the survival of the fittest mechanism in nature. It is expected that a fitter chromosome receives a higher number of offspring and thus has a higher chance of surviving in the subsequent evolution while the weaker chromosomes will die eventually. The Roulette wheel selections one of the most common and easy-to-implement selection mechanism. A virtual wheel is used in this selection mechanism. Each chromosome in the population is associated with a sector in the virtual wheel. According to the fitness value of the chromosome, the sector will have a larger area when the corresponding chromosome has a better fitness value while a lower fitness value will lead to a smaller sector. In order to implement the Roulette wheel selection for GA, we will use Eq.(4) to normalize the fitness values of the chromosomes.

$$p_i = F_i / \sum_{i=1}^{M} F_i, \quad i=1,2,\cdots,M \tag{4}$$

Where P_i is the rormalized fitness value of Mth chromosome in the population and F_i is the fitness value of chromosome in the population.

4) Crossover

Crossover operator is a recombination operator that combines subparts of the parents to produce offspring that contain some parts of both parent genetic materials. The selected parents most likely are the fitter chromosomes. Therefore, it can be seen that this operator is aimed to combine the optimized genetic materials in the parents together and to produce more optimized offspring. In this operator, two parents will be selected from the population pool based on the Roulette wheel mechanism. The crossover formulas are shown as follows:

$$X_A^{t+1} = aX_B^t + (1-a)X_A^t \tag{5}$$

$$X_B^{t+1} = aX_A^t + (1-a)X_B^t \tag{6}$$

Where a =0.5.

5) Mutation

Mutation adds variations of model parameters into chromosomes. It provides the global searching capability to our GA by randomly altering the values of genes in the chromosomes. It recovers the lost information in the initialization phase of our GA and makes our GA escape from the initial model parameters and to find the optimal model parameter set. According to an alternation probability, each model parameter in the parent may or may not be altered by this genetic operator. Before the modification of a model parameter, the mutation rate will be compared with a randomly generated probability to test if the mutation rate is larger than or equal to the randomly generated probability. If the result is true, then the model parameter is altered by the equation

$$change = (1-p) \times (random \times (1 - \frac{ct}{mt}))^b \tag{7}$$

Where b=2.

6) Hybrid operations

As mentioned in previous sections, we used the hybrid-GA to improve the convergence speed of the GA-based HMM training. Unlike the traditional GA, our hybrid-GA will apply the Baum-Welch algorithm with eight iterations to the chromosomes in the population pool every 20 generations such that the fitness value of each chromosome is improved. In addition, we also apply the re-estimation procedure to the offspring in the subpopulation. In order to make the replacement scheme work properly, we will apply the Baum-Welch algorithm with three iterations to the offspring and compare it with the chromosomes in the population pool.

7) Information extraction

GA-HMM may be used for web information extraction by formulating a model in the following way: each state is associated with a class that we want to extract, such as title, author or booktitle. Each state emits words from a class-specific unigram distribution. We can learn the class-specific unigram distribution and the stata transition probabilities from training data by GA-HMM hybrid operations. In order to label a new web with classes, we treat the words from the web as observations and recover the most-likely state sequence with the Viterbi algorithm. The state that produces each word is the class tag for that word.

4. Experimental results

In this section, we evaluate the experimental results of our system that includes two modules: HMM and GA-HMM. The training and testing data for our system are also available at www.jaist.ac.jp/~hieuxuan/softwares/pewebme/. They include approximately 2100 webs. 95% of this set was chosen randomly for training and the rest is the testing set.

The training and testing data process is as follows. First, the input web page is parsed to build an HTML tree. Then, we locate data regions containing data records by estimating the shannon's entropy at each internal node. Found records are transformed into sequences of data segments. Training data and testing data are a set of sequences of segments. Each sequence corresponds to a data record, and each segment belongs to one state. All sequences in the data set are labeled as title, author, booktitle, publisher, and year.

Various kinds of instance with different GA control parameters have been solved with our algorithm. The results are listed in Table 1, we ran each instance 20 times with a different initial population size, crossover rate, and mutation rate, and obtained the average P[o|λ] values of the 20 executions.

Population size	Crossover rate	Mutation rate	Average P[o|λ]
50	0.1	0.01	6.849923
50	0.6	0.03	6.962669
50	0.8	0.1	6.499843
100	0.1	0.01	7.168990
100	0.6	0.03	7.418719
100	0.8	0.1	7.335336
200	0.1	0.01	8.082554
200	0.6	0.03	6.943885
200	0.8	0.1	6.828164

Table 1: GA parameters training HMM.

We can see that population size is 200, crossover rate is 0.1, and mutation rate is 0.01 are superior to all the other approaches in all cases. Therefore, in our GA-HMM, we use the following control parameters:

Population size: 200
Crossover rate: 0.1
Mutation rate: 0.01

Our GA evolution will terminate after 60 generations.

The experimental results comparing GA-HMM and HMM for web information extraction are described in Table 2.The first column is the states. The second and third columns are the recall and the precision for HMM module. The next two columns are similar experimental results for GA-HMM module. The average is calculated from the sum of 5 recall and precision. Generally speaking, average reflect the extent to which a system can adapt to different web structures and the total performance of a system with respect to the total number of webs that are correctly extracted. In the GA-HMM column, the values indicated the optimal number of states found by our GA-HMM training. Our GA-HMM training is able to find the optimal number of states in all the cases of 2100 webs. Besides, the HMM trained by our GA-HMM training have higher value than the HMM trained by the Baum-Welch algorithm. This implies that the HMM trained by our GA-HMM method are more optimized than the HMM trained by the Baum-Welch method. The average precision rates for GA-HMM and HMM is 84.483% and 71.049%, respectively. Therefore, it can be concluded that a better training always leads to a better extraction.

States	HMM		GA-HMM	
	recall	precision	recall	precision
Title	0.75308	0.46585	0.87655	0.70647
Author	0.83870	0.85714	0.89247	0.84298
Booktitle	0.58261	0.80723	0.68696	0.95238
Publisher	0.64150	0.75556	0.77359	0.85417
Year	0.81818	0.66667	0.90910	0.86813
Average	0.72681	0.71049	0.82773	0.84483

Table 2: Experimental results of GA-HMM and HMM.

5. Conclusions and future work

In this paper, we have presented a new HMM training method based on genetic algorithms for web information extraction. This method finds good HMM topology as well as its model parameters. From our experiments with the 2100 webs extracted from our corpus, our method is able to find the optimal states in all cases. In addition, the HMM trained by our GA-HMM training have higher values of precision and recall than the HMM trained by the Baum-Welch algorithm. This implies that the HMM trained by our GA-HMM method are more optimized than the HMM trained by the Baum-Welch method. It can be concluded that the number of states in each word model and the model parameters of the HMM are optimized in one single step.

From the experimental results, our genetic operator can find the optimal number of states even in the case of uneven initial distribution of the number of occurrences of the chromosomes with different number of states. This indicated that our operator is a robust approach for finding the optimal number of states in HMM. We also find that GA is a very easy-to-use optimization algorithm. By using a very simple genetic operator, our GA can find the optimal number of states in the word model efficiently.

Future work will try to enhance the precision and recall for our system. Feature selection will be refined by using both human perceptions and feature selection algorithms. The use of dictionaries for unseen words or phrases may be an efficient way to reduce classification errors. Finally, more natural language processing techniques such as text chunking and named entity recognition will be employed to preprocess data and make the system more intelligent.

Acknowledgement

This work is supported by Hunan Natural Science Foundation (Grant No. 04JJ40051) and Scientific Research Foundation of Hunan Provincial Education Department (Grant No. 06c724).

References

[1] D. Freitag and A. Mccallum, Information extraction with HMMs and shrinkage. *Proceedings of the AAAI'99 Workshop on Machine Learning for Information Extraction,* pp.31-36, 1999.

[2] D. Freitag and A. Mccallum, Information extraction with HMM structures learned by stochastic optimization. *Proceedings of the Eighteenth Conference on Artificial Intelligence,* pp.584-589, 2000.

[3] K. Seymore , A. Mccallum and R. Rosenfeld, Learning hidden markov model structure for information extraction. *AAAI'99 Workshop on Machine Learning for Information Extraction,* pp.37-42, 1999.

[4] D. Freitag , A. Mccallum and F. Pereira, Maximum entropy markov models for information extraction and seqmentation. *Proceedings of ICML-2000*, pp.591-598, 2000.

[5] D. Bouchaffra and J. Tan, Structural hidden markov models using a relation of equivalence:

application to automotive designs. *Data Mining and knowledge Discovery*,12:79-96, 2006.

[6] R.J. Mooney and U.Y. Nahm, Text mining with information extraction. Multilingualism and Electronic language Management, *Proceedings of the 4th International MIDP Colloquium,* pp.141-160, 2005.

[7] X.H. Phan, S. Horiguchi and T.B. Ho, Automated data extraction from the web with conditional models. *Int.J. Business Intelligence and Data mining,* 2:191-209, 2005.

[8] S. Kwong, C.W. Chan, K.F. Man and K.S. Tang, Optimization of HMM topology and its model parameters by genetic algorithms, *Pattern Recognition,*34:509-522, 2001.

[9] Q.Y. Hong and S. Kwong, A genetic classification method for speaker recognition. *Engineering Applications of Artificial intelligence*,18:13-19, 2005.

[10] A. Asllani and A. Lari, Using genetic algorithm for dynamic and multiple criteria web-site optimizations. *European Journal of Operational Research,*176:1767-1777, 2007.

[11] M. Caramia, G. Felici and A. Pezzoli, Improving search results with data mining in a thematic search engine. *Computers & operations Research,*31:2387-2404, 2004.

[12] H. Zhon, Y.C. Feng and L.M. Han, The hybrid heuristic genetic algorithm for job shop scheduling. *Computers & Industrial Engineering,*40:191-200, 2001.