An Empirical Exploration of Hidden Markov Models: From Spelling Recognition to Speech Recognition

Shieu-Hong Lin
Department of Mathematics and Computer Science
Biola University
13800 Biola Avenue
La Mirada, California 90639
shieu-hong.lin@biola.edu

Abstract

Hidden Markov models play a critical role in the modelling and problem solving of important AI tasks such as speech recognition and natural language processing. However, the students often have difficulty in understanding the essence and applications of Hidden Markov models in the context of a cursory introductory coverage of the subject. In this paper, we describe an empirical approach to explore the subject of the Hidden Markov models. This approach focuses on a series of incremental developments of Hidden Markov models for automatic spelling recognition. The process of programming and experiments with these models cultivates the actual modelling and problem-solving capacity, and guides the students to a better understanding of the application of similar Hidden Markov models used in speech recognition.

1 Introduction

The mathematical framework of the Hidden Markov models (HMMs) along with the algorithms related to HMMs have played a very important role in the successful development of intelligent systems for a wide variety of AI tasks in speech recognition (Jelinek 1998) (Russell & Norvig 2003) and natural language processing (Charniak 1994) (Manning & Schütze 1999). In AI courses, automatic speech recognition is often cited as a challenging and yet well accomplished AI task by using HMMs and probabilistic reasoning (Russell & Norvig 2003). To illustrate the general mechanism of modern speech recognition systems, the mathematical definition of HMMs is provided along with description of the development of the underlying acoustic model as an HMM (Russell & Norvig 2003). This approach only presents a somewhat abstract theoretical framework of the use of HMMs, and the students do not have a chance to be involved in the process of modelling and problem solving using HMMs.

We believe that a well designed course project about HMMs is needed to better motivate the students to learn HMMs and cultivate the modelling and problem solving skills transferrable to other challenging AI tasks. The use of course projects and its benefits have been well acknowledged in the general context of education (Blumenfeld & Soloway 1991) (Barron *et al.* 1998) and in teaching specific AI subjects such as stochastic local search (Neller 2005) and case-based reasoning (Bogaerts & Leake 2005).

The main challenge to this project-based empirical approach of teaching HMMs is how to tailor the project to retain enough details and make it real and thought-provoking, while in the meantime avoid too much excursion outside the subject of HMMs. For example, an empirical exploration of speech recognition using real-world acoustic data would require significant excursion to bring to an in-depth understanding of at least (i) the processing of acoustic signals into frames and the data format of these acoustic frames, and (ii) the actual linguistic data for phone models and pronunciation models of real words (Russell & Norvig 2003). To enable real-time interactions between the students and the systems they are building, it would also incur additional overhead of hardware and software setups to collect and analyze acoustic data. In this regard, speech recognition is not the best choice to be the center of the hands-on project on HMMs.

In this paper, we describe the framework of a hands-on project using HMMs for automatic recognition of misspelled words. There is no complication in explaining the concept of spelling recognition, the acquiring of data, or the real-time interaction between the students and the systems they are building. Both automatic spelling recognition (Jurafsky & Martin 2000) and automatic speech recognition (Jelinek 1998) can be viewed as tasks of decoding corrupted messages received from a noisy-channel (Mackay 2003), and the HMMs used for spelling recognition in the project are structurally parallel to those used in speech recognition. This allows us to conveniently find analogy to explain the general mechanism of modern speech recognition systems based on the experiences with the spelling recognition systems the students are building. The framework can be flexibly tailored from a couple of lab and programming assignments to an extensive semester project. The project website at http://csci.biola.edu/HMM provides additional details.

2 Chat-Room Spelling Recognition

In the chat-room environment, the participants under their pseudo names can communicate with one another by exchanging text entered from the keyboard. Because of cognitive and typographic mistakes, character strings appearing in the text inputs in the chat room may not be valid words in the vocabulary, and we have to guess what the actual intended words are. For example, when the string "*iis*" appears in the text input from a participant, we have to figure out the most likely words in the vocabulary such as "*is*" and "*its*" as the source of the corrupted string. When we are familiar with the patterns of corrupted words from a frequent participant, we may well recognize the original message from the corrupted text inputs. Even when the frequent participant p is under the disguise of a new pseudo name, we may be able to recognize this seemingly unknown person by analyzing the sample text inputs and its consistency with the spelling characteristics of the well known participant p.

To highlight the analogy to speech recognition, we intentionally use the term *spelling recognition* instead of spelling correction. We focus on spelling recognition for English and assume the text inputs are case-insensitive. In section 3, we describe very simple probabilistic models for characterizing how chat-room participants may (erroneously or not) enter words through the keyboard. Given (i) the probabilistic models for text inputs through the keyboard and (ii) a vocabulary set V of words in Σ^* with respect to the English alphabet Σ, we formulate the following spelling recognition tasks in the context of the chat-room environment.

Single-word spelling recognition: Given a character string x entered by a specific participant p, the task of single-word spelling recognition is to determine the top k most likely words in V that could be (erroneously) typed by the participant p as the character string x, where k is either one or a small integer.

Multi-word spelling recognition: Given a sequence of c character strings $\mathbf{X} = \langle x_1, x_2, \ldots x_c \rangle$ entered by a specific participant p, the task of multi-word spelling recognition is to determine the most likely sequence of c words that may end in the sequence of character strings $\mathbf{X}$ when entered by the participant p.

Participant identity recognition: Given a sequence f c character strings $\mathbf{X} = \langle x_1, x_2, \ldots x_c \rangle$ entered by an unidentified participant, the task of participant identity recognition is to determine, among a set of well known participants, the most likely participant that may generate the strings in $\mathbf{X}$.

3 Modelling the Input Process of Words

To be able to conduct automatic spelling recognition, we need to devise probabilistic models of how participants input individual words through the keyboard. As the first step of the project, we define a very simple parameterized *spelling model* and a very simple parameterized *keyboard model* for this purpose, from which we can construct HMMs used in probabilistic reasoning for spelling recognition.

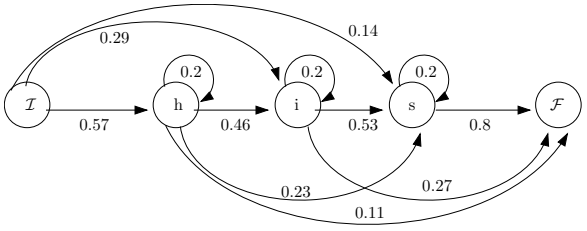

Figure 1: The spelling model regarding the word "*his*" with the parameters $deg_{sp} = 2$, $p_{repeat} = 0.2$, and $p_{moveOn} = 0.8$

3.1 The Spelling Model

The main purpose of the spelling model is to model the possibilities of spelling mistakes such as repeating or skipping characters when a participant enters a word as a sequence of characters.

Probabilistic state-transition diagrams: For each distinct participant p and a given word w, the spelling model employs a probabilistic state-transition diagram $\mathcal{SP}_p^w$ defined in the following to describe the possible transitions of cognitive states and the associated probabilities when w is entered as a sequence of characters by p. See the example in Figure 1 and the definitions of the parameters involved later in Section 3.1.

States and the order of states in the diagrams: For each distinct participant p and a given word w, the states in $\mathcal{SP}_p^w$ are ordered and represented as nodes listed from left to right in the diagram. The initial state $\mathcal{I}$ is the first state, which models p's cognitive state of acknowledging the beginning of inputting w. Then one by one for each character c in the word w, there is a corresponding *character state* in $\mathcal{SP}_p^w$, which models p's cognitive state of commitment to typing that particular character c. In the end, the final state $\mathcal{F}$ models p's cognitive state of acknowledging the end of inputting w. This gives us $n+2$ distinct states in total for a word w of n characters.

State transitions and spelling mistakes: Possible state transitions are represented as arcs in the diagram as shown in Figure 1. From the initial state $\mathcal{I}$, it is possible to transition into to any character state. From each character state, it is possible to transition into either the same state again, or to any later character state (i.e. character states to its right), or to the final state. Each time the participant p goes through the process of inputting the word w, a series of cognitive state transitions occurs, starting from the initial state $\mathcal{I}$ and ending in the final state $\mathcal{F}$. If the word w is spelled out per-

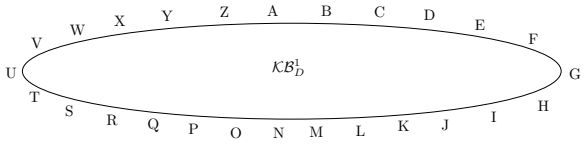

Figure 2: The one-dimensional keyboard model $\mathcal{KB}^1_D$

fectly without mistake, it should start from the initial state, go through the character states one by one in the order, and end in the final state. However, spelling mistakes may occur and end in repetition and/or skipping of some character states in the series of state transition.

Transition probabilities: In general, the probability of a state transition from a state s to a state t can be any real number in the range of $[0, 1]$ and is associated with the arc from s to t in the diagram. The only constraint is that the sum of transition probabilities from a state s must be one for every state s other than the final state.

Parameters for modelling cognitive mistakes: To ease the initial programming tasks for the students, we allow the use of three parameters deg_{sp}, p_{repeat}, and p_{moveOn} to further simplify the spelling model. The parameter deg_{sp} is the rate of exponential degrading of probability with respect to the number of character skipped. The participant may skip one or more characters, but the probability of skipping d_1 characters is $deg_{sp}^{-(d_1-d_2)}$ times the probability of skipping d_2 characters. For each character state, p_{repeat} is the probability to transition into the same state again while p_{moveOn} is the sum of probabilities to transition into any later character state or to the final state (i.e. the states to its right). Obviously, the sum of p_{moveOn} and p_{repeat} must be one.

Calculating the transition probabilities automatically: Note that (i) the sum of transition probabilities from each specific state (other than the final state) into new states must equal p_{moveOn} and (ii) the relative scales of transition probabilities from each specific state into new states are completely determined by deg_{sp}. This allows us to easily write code to automatically calculate the probabilities of all possible transitions in the diagram $\mathcal{SP}^w_p$ based on the parameters deg_{sp} and p_{moveOn}.

3.2 The Keyboard Model

The main purpose of the keyboard model is to model the probabilities of typographic mistakes of pressing a wrong key different from the intended character targeted by the current cognitive state.

Probabilistic parameters for typographic mistakes: To ease the initial programming tasks for the students, we use three probabilistic parameters p_{hit}, p_{miss}, and deg_{kb} to simplify the keyboard model. The parameter p_{hit} is the probability that the exact key for the intended character is pressed to generate the character. The parameter p_{miss} is the probability that a wrong key is pressed. Obviously, the sum of p_{hit} and P_{miss} must equal one. The parameter deg_{kb} is the rate of exponential degrading of probability with respect to the distance between the key of the intended character and the key wrongly pressed. The probability of hitting a wrong key that is in a distance of d_1 away from the intended key is $deg_{kb}^{-(d_1-d_2)}$ times the probability of hitting a wrong key that is in a distance of d_2 distance away from the intended key. Two simple distance metrics $\mathcal{KB}^1_D$ and $\mathcal{KB}^2_D$ are considered in the following.

One-dimensional keyboard distance metric $\mathcal{KB}^1_D$: Imagine a special keyword in the shape of a circle as shown in Figure 2 with the keys for the twenty six English letters evenly arranged on the keyboard in the alphabetical order. For example, 'A' is immediately adjacent to 'Z' and 'B', 'B' is immediately adjacent to 'A' and 'C',. . ., and 'Z' is immediately adjacent to 'Y' and 'A'. The distance between two keys is simply one plus the minimum number of keys separating them. For example, the distance between 'A' and 'B' is one, the distance between 'A' and 'C' is two,. . ., the distance between 'A' and 'Y' is two, and the distance between 'A' and 'Z' is one,

Two-dimensional keyboard distance metric $\mathcal{KB}^2_D$: For the standard keyboard we use, the keys for the twenty six English letters are arranged in three rows. There are many possible metrics to measure the distance between keys. However, for simplicity $\mathcal{KB}^2_D$ identifies the location of the jth key on the ith row as (i, j), and uses the standard Euclidean distance as the distance metric.

Calculating the probabilities of typographic mistakes: Note that (i) for each specific character c the sum of probabilities of pressing wrong keys must equal p_{miss} and (ii) the relative scales of the probabilities of pressing these wrong keys are completely determined by deg_{kb} and the selected keyboard distance metric. This allows us to easily write code to automatically calculate the probabilities of pressing wrong keys given the intended key based on the parameters deg_{kb}, p_{miss} and the selected keyboard distance metric.

3.3 The Language Model and Further Extensions

For simplicity, our basic framework implicitly uses the null language model, which assumes that each word is equally likely to be used regardless of the context and therefore sentences of the same length are equally likely. It is possible to adopt a more sophisticated language model such as the bigram language model (Charniak 1994) (Manning & Schütze 1999). See section 4 about how this would allow us to take advantage of the context of the observed strings when conducting multi-word spelling recognition. Also see section 5.2 for further extensions of the spelling model and the keyboard model.

4 Using HMMS for Spelling Recognition

The spelling model for spelling recognition describes the dynamics of transitions in cognitive states. Although we cannot directly observe the underlying cognitive state, we can observe the character generated as a result from a cognitive state of trying to press a particular key. And the keyboard model describes the probabilistic correlation between the actual character generated and intended key to press. For the spelling-recognition tasks in section 2, we have to reason about the most likely words or persons based on a sequence of observable characters (as corrupted words) generated by a sequence of hidden underlying cognitive states. In the following, we show how we can well accomplish the tasks by using the hidden Markov models and applying the well known *forward-backward algorithm* to calculate likelihood (Manning & Schütze 1999) (Jurafsky & Martin 2000) (Russell & Norvig 2003).

HMMS for Spelling Recognition: Mathematically, a hidden Markov model is a four tuple $\langle \mathbf{S}, \mathbf{R}, \mathbf{T}, \mathbf{O} \rangle$ where **S** is a finite set of underlying states $\{s_1, s_2, \ldots, s_n\}$, **R** is a finite set of observable results $\{r_1, r_2, \ldots, r_m\}$, **T** is a transition probability matrix with the element T_{ij} recording the probability of a state transition from the current state s_i to the next state s_j, and **O** is an observation-likelihood probability matrix with the element O_{ij} recording the probability of observing the result r_j when the underlying system state is s_i. For our spelling recognition tasks, the combination of the spelling model and the keyboard model together for a particular chat-room participant p ends in a collection of hidden Markov models: $\mathcal{HMM}_p^w = \langle \mathbf{S}, \mathbf{R}, \mathbf{T}, \mathbf{O} \rangle$ for each word W in the vocabulary set V where (i) the state set **S** is simply the set of states in the state transition diagram $\mathcal{SP}_p^w$ of the spelling model in section 3.1, (ii) the observable result set **R** is simply the English alphabet Σ, composed of all the English letters, (iii) the transition probability matrix **T** is determined by the transition probabilities in the state transition diagram $\mathcal{SP}_p^w$ of the spelling model in section 3.1, and (iv) the observation-likelihood probability matrix **O** is determined by the probabilities of typographic mistakes of the keyboard model in section 3.2.

Accomplishing the Spelling Recognition Tasks:
Given a hidden Markov model m of n states and a sequence of results **r** observed, the well known forward-backward algorithm can determine $Pr(\mathbf{r}|m)$, the probability of observing **r** in $O(|\mathbf{r}|n^2)$ time while the Viterbi algorithm can determine the most likely underlying state sequence in $O(|\mathbf{r}|n^2)$ time (Manning & Schütze 1999) (Russell & Norvig 2003). For spelling recognition, a sequence of results **r** simply means a sequence of characters as a corrupted or uncorrupted word entered by a chat-room participant, and we have to consider the collection of hidden Markov models $\mathcal{HMM}_p^w$ as the combination of the spelling model and the keyboard model for each chat-room participant p and each word w in the vocabulary set V. In the following, we show how we can accomplish the spelling recognition tasks by applying the forward-backward algorithm and the Viterbi algorithm to the hidden Markov models $\mathcal{HMM}_p^w$ described above.

Algorithm for single-word spelling recognition: Given a specific chat-room participant p and an observed character string x, we apply the forward-backward algorithm for each word w in the vocabulary set V to calculate $Pr(x|\mathcal{HMM}_p^w)$, the probability of seeing x when the participant p actually intends to type the word w. The top k most likely words for the character string x are simply those words associated with the highest k probabilities we got.

Algorithm for multi-word spelling recognition: Given a specific chat-room participant p and a sequence of c observed character strings $\mathbf{X} = \langle x_1, x_2, \ldots, x_c \rangle$, if the default null language model is used, we simply apply single-word spelling recognition to determine the single most likely word $w_p^{x_i}$ for each string x_i in the sequence **X** and the most likely sequence of words is simply $\mathbf{W}_p = \langle w_p^{x_1}, w_p^{x_2}, \ldots, w_p^{x_c} \rangle$. Instead, if the bigram language model is adopted for modelling the likelihoods of sequences of words, we should for each observed character strings keep the most likely k candidate words (for a small k) and apply the well known Viterbi algorithm to pick the single most like sequence of words out of the candidates.

Algorithm for participant identity recognition: Given a sequence of c observed character strings $\mathbf{X} = \langle x_1, x_2, \ldots, x_c \rangle$ and a set of d well known participants $\mathbf{P} = \{p_1, p_2, \ldots, p_d\}$, we can apply multi-word spelling recognition to determine the most likely word sequence $\mathbf{W}_{p_i} = \langle w_{p_i}^{x_1}, w_{p_i}^{x_2}, \ldots, w_{p_i}^{x_c} \rangle$ corresponding to each participant p_i in **P**. In the meantime, for each participant p_i, we can calculate $\prod_{x \in X} Pr(x|\mathcal{HMM}_{p_i}^{w_{p_i}^x})$, the probability of seeing the sequence of character strings **X** entered by p when the word sequence $\mathbf{W}_p$ is actually intended by p. As a heuristic, the participant p with the highest such probability value is picked as the most like participant to generate the sequence of strings **X**.

5 Framework of Empirical Exploration

We implement a simulated interactive chat-room environment for spelling recognition as the foundation for further empirical exploration. Through a series of hands-on lab experiments and programming assignments on top of the simulated environment, the students are guided to develop a solid understanding of the essence and applications of HMMs.

5.1 Simulating the Chat-Room Environment

We implement the key constructs such as the chat room, the participants, the spelling model, and the keyboard model as classes. Objects of these classes are then initiated to sim-

ulate the chat-room environment. The implementation can be done in object-oriented programming languages such as Java and C++.

The chatroom class: This class provides the simulation of the chat-room environment. Each *chatroom* object contains a dynamic array of *participant* objects and a collection of words as the vocabulary set, and provides methods for the outside world to interact with the participants.

The Participant class: Objects of this class simulate the behaviors of chat-room participants. Each *participant* object contains a *keyboardModel* object and a *spellingModel* object in it, and provides methods to (i) simulate the series of characters that may actually appear as the result of trying to enter a given word, (ii) explain the cognitive and typographic mistakes that end in the series of characters that appear as the result of trying to enter a given word.

The KeyboardModel class: This is an abstract class with two subclasses *keyboard1D* and *keyboard2D* to simulate the behavior of typing using the keyboard distance metrics $\mathcal{KB}_D^1$ and $\mathcal{KB}_D^2$ respectively regarding the keyboard model described in section 3.2. An object of this class is initialized with the probabilistic parameters for typographic mistakes p_{hit}, p_{miss}, and deg_{kb} as its attributes, and provides methods to (i) simulate what the actual character may be seen as the result of the chat-room participant trying to press a given target key on the keyboard and (ii) calculate the probability of seeing a character c when the participant intends to press a given target key based on the probabilistic parameters p_{hit}, p_{miss}, and deg_{kb}.

The SpellingModel class: This class provides the simulation of flow of cognitive states (as described in section 3.1 regarding the spelling model) when the chat-room participant tries to enter a word as a series of characters. An object of this class is initialized with the probabilistic parameters for cognitive mistakes deg_{sp}, p_{repeat}, and p_{moveOn} as its attributes, and provides methods to (i) simulate what may be the next cognitive state of the chat-room participant given the current cognitive state in the process of entering a given word and (ii) calculate the probability of a state transition from one cognitive state to another based on the probabilistic parameters deg_{sp}, p_{repeat}, and p_{moveOn}.

The SpellingRecogHMMs class:
Each *SpellingRecogHMMs* object contains a *keyboardModel* object and a *spellingModel* object and serves as an abstract representation of the HMMs for speech recognition concerning the *keyboardModel* object and the *spellingModel* object. It also provides a method that can (by applying the forward-backward algorithm to the underlying HMM $\mathcal{HMM}_p^w$) determine $Pr(x|\mathcal{HMM}_p^w)$, the probability of seeing x when a participant p with the corresponding *keyboardModel* object and *spellingModel* intends to type the word w.

5.2 Experiments and Programming Assignments

We use the simulated chat-room environment as the test ground for the students to conduct experiments to get familiar with the spelling-recognition domain, to implement their own code to accomplish automatic spelling recognition, to debug the code, and to evaluate the accuracy of spelling recognition achieved.

Initiate the chat-room environment: We create a chat room with two participants with quite different typing behaviors. The participants use different values for the parameters in the spelling model and the keyboard model. One of the participants uses the one-dimensional keyboard model while the other uses the two-dimensional keyboard model.

Interact with the chat-room participants: We give the students a limited number of words and the students need to ask each chat-room participant to (i) type each of these words several times and (ii) explain the underlying cognitive and typographic mistakes. The purpose of this experiment is to have students directly observe the correlation of between (i) the different models and the parameter values used by the participants and (ii) the ways text inputs are corrupted by the participants.

Conduct spelling recognition of words manually: For each chat-room participant, we provide a small sample text and ask the participant to enter the text. We then collect the results as corrupted texts, label the corrupted texts with the names of the corresponding participants, and ask the students to manually recover the original text messages. In the end, the students compare what they have recovered to the original texts and evaluate the accuracy they achieve.

Conduct participant identity recognition manually: In this experiment, we collect corrupted texts generated by the participants just like in the previous experiment, but we do not label them with names of the corresponding participants. The students have to first try to recognize the participant behind each corrupted text. The actual participants behind the corrupted texts are then revealed to the students to evaluate the accuracy they achieve

Implement automatic spelling recognition: After the experiments above, the students are already familiar with spelling recognition domain. We then ask the students to implement the spelling-recognition algorithms described in section 4.2 on top of the implementation of the *SpellingRecogHMMs* class.

Conduct automatic spelling recognition: At this point, the students can conduct automatic spelling recognition of words and participant identity recognition by applying their code to corrupted texts they have played with in the earlier experiments. The students can compare the accuracy achieved by their code with that achieved by people to see the power of automatic spelling recognition using HMMs.

Further extensions of the framework: The simple spelling model and the simple keyboard in section 3.2 can be enhanced to extensively cover cognitive mistakes such as transposing the order of two adjacent characters in the word (Jurafsky & Martin 2000) and the general edit operations described in (Brill & Moore 2000). Another useful extension of the basic framework is to learn the state-transition probabilities and the observation-likelihood probabilities of the underlying HMMS directly from the corpus of spelling errors (Manning & Schütze 1999) (Brill & Moore 2000), instead of parameterizing the spelling model and the keyboard and setting the parameter values manually. These extensions can easily end in a semester project for advanced students to substantially expand their implementation of the basic framework to (i) incorporate the bigram language model as described in section 3.3, (ii) refine the spelling model, and (iii) to learn probabilities from the spelling-error corpus.

6 From Spelling Recognition to Speech Recognition

The following is a brief highlight of the analogy between (i) the roles of spelling model, the keyboard model, the language model, and the HMMs used in spelling recognition and (ii) the roles of the *pronunciation model*, the *phone model*, the language model, and the HMMs used in speech recognition (Jurafsky & Martin 2000) (Russell & Norvig 2003). First of all, note the similarity between the process of typing words and the process of pronouncing words. The chat-room participant enters a word by transitions through a series of cognitive states of typing individual characters. For speech recognition the speaker pronounces a word by transitions through a series of cognitive states to pronounce phones. As an analogy to the spelling model in spelling recognition, the *pronunciation model* is used in speech recognition to model for each word the dynamics of possible transitions of the speaker's cognitive states underlying the phones to pronounce. The keyboard model in spelling recognition models the likelihoods of possible characters being observed as the result of the participant intending to press a specific key. As an analogy, the *phone model* in speech recognition models the likelihoods of various possible acoustic patterns being observed as the result of the speaker intending to pronounce a specific phone. In both spelling recognition and speech recognition, exactly the same kind of models such as the bigram model can be used as the language model to better differentiate the likelihoods of words given different context of neighboring words. Parallel to how HMMs are derived from the spelling model and the keyboard model for spelling recognition, HMMs are derived from the pronunciation model and the phone model. In both spelling recognition and speech recognition, the forward-backward algorithm and the Viterbi algorithm are then applied to these HMMs to recover the most likely individual words and/or word sequences.

Because of the analogy between spelling recognition and speech recognition, students can naturally transfer what they have learned from the empirical exploration of spelling recognition to facilitate a concrete understanding of the application of HMMs in modern speech recognition systems.

7 Conclusion

Hidden Markov models play a critical role in the modelling and problem solving of important AI tasks such as speech recognition and natural language processing. However, a cursory theoretical coverage of the subject can rarely establish a concrete understanding of the essence and applications of Hidden Markov models. In this paper, we describe an empirical approach to explore the subject of the Hidden Markov models. This approach analyzes the text-input process and develops Hidden Markov models for automatic spelling recognition. Through a series of incremental programming and experimental assignments, we can cultivate the actual modelling and problem-solving capacity of the students, and guides the students to an in-depth understanding of the application of similar Hidden Markov models used in speech recognition.

References

Barron, B. J.; Schwartz, D. L.; Vye, N. J.; Moore, A.; Pertrosino, A.; Zech, L.; and Bransford, J. 1998. Doing with understanding: lessons from research on problem- and project-based learning. *Journal of the Learning Sciences* 7:271–311.

Blumenfeld, P. C., and Soloway, E. 1991. Motivating project-based learning: sustaining the doing, supporting the learning. *Educational Psychologist* 26:369–398.

Bogaerts, S., and Leake, D. 2005. Increasing ai project effectiveness with reusable code framework: a case study using iucbrf. In *Proc. of FLAIRS-2005*. AAAI Press.

Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *Proc. of the 38th Annual Meeting of the ACL*, pages 286–293.

Charniak, E. 1994. *Statistical Language Learning*. MIT Press.

Jelinek, F. 1998. *Statistical Methods for Speech Recognition.* MIT Press.

Jurafsky, D., and Martin, J. H. 2000. *Speech and Lnaguage Processing*. Prentice Hall.

Mackay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge Universirt Press.

Manning, C. D., and Schütze, H. 1999. *Foundations Of Statistical Natural Language Processing*. MIT Press.

Neller, T. W. 2005. Teaching stochastic local search. In *Proc. of the FLAIRS-2005*. AAAI Press.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach,*. Prentice Hall, 2nd edition.