

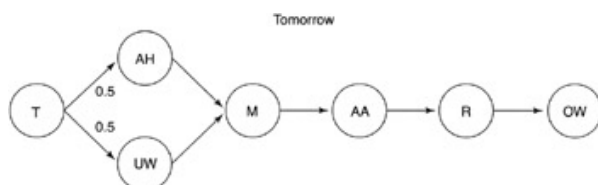
## Chapter 10: The Bigram Model

 Download CD Content

In this chapter, we'll look at the **Markov model** (named after the Russian mathematician, Andrei Markov), and a specific variation called the **bigram model**. These models can be very useful in describing processes that encompass a number of states and the probabilities associated with paths through these states. A number of very interesting applications exist for these models. We'll investigate the use of the bigram model for the automatic generation of interesting text.

## Introduction to Markov Models

A Markov Chain is simply a process that consists of a number of states with probabilities attached to the transitions between the states. For example, consider [Figure 10.1](#).

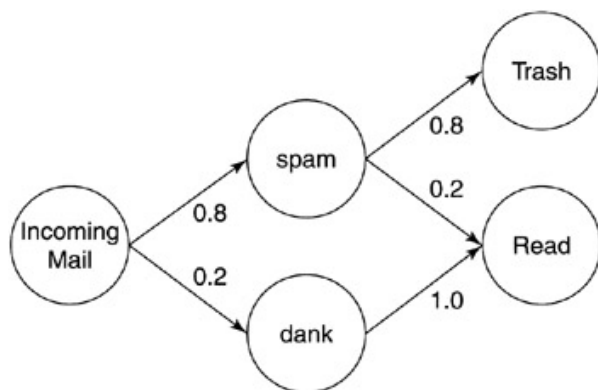


**Figure 10.1:** Example Markov Chain.

[Figure 10.1](#) illustrates the pronunciation of the word "tomorrow." Two possible pronunciations are available within the diagram. The probability of the "tahmorrow" pronunciation is 0.5, while the probability of the alternate "tuwmorrow" pronunciation is also 0.5.

This is a very simple case, involving a decision at a single point within the chain. Each state involves the production of a phoneme. At the end of the chain, a completed pronunciation is available. We'll look more at the purpose for this in the applications section.

Next, let's look at a different application. Consider an email program that monitors the behavior of the user. When an email arrives, it observes what the user does with the email, and uses this information to learn how to automatically deal with subsequent emails. See the chain in [Figure 10.2](#)



**Figure 10.2:** Learning user interaction with an email program.

Our email agent has observed that 8 out of 10 emails are spam, 2 out of 10 are from user "dank." The email agent further observes that 80% of the time, we delete the spam email without reading it. The other 20%, we read the email. From these probabilities, the email agent can deduce that we're more likely to delete the email than to read it. Using these probabilities provides an opportunity for the email agent to simplify our task of managing email.

**Tip** In the examples shown here, a limited number of states and connections were defined. The Markov Chain can support a very large state space with very large numbers of connections to model very complex processes.

An interesting property of both examples is that the current state is always a function of the previous state, given a connection probability. This is called the "Markov Property." Additionally, since in our examples, a state can be reached by more than one previous state (for example, the "Read" state from [Figure 10.2](#)), these models are known as Hidden Markov Models (HMMs) or Hidden Markov Chains.

## HMM Approximations

Note that in the prior examples, the current state of the chain was a function solely of the prior state of the chain with a defined probability. This is known as a **bigram** (a sequence of two words). If the current state were not a function of the prior state, the selection of state would simply be a random process. If the state were dependent upon the prior two states, it would be called a **trigram**. While increasing the dependence on the prior states (or context) can increase the usability of the chain (as we'll see in the applications section), the memory requirements on such models can be inhibitive [Henke 1999]. For example, consider Table 10.1 that shows the number of elements necessary for a vocabulary of 100 words.



**Table 10.1: Context Size and the Number of Unique Elements.**

<i>n</i>	<i>Type</i>	<i>Number of Elements</i>
2	bigram	10,000
3	trigram	1,000,000
4	4-gram	100,000,000



A corpus of 100 unique words is quite small, therefore creating anything other than bigrams over a corpus can be quite expensive.

## Interesting Applications

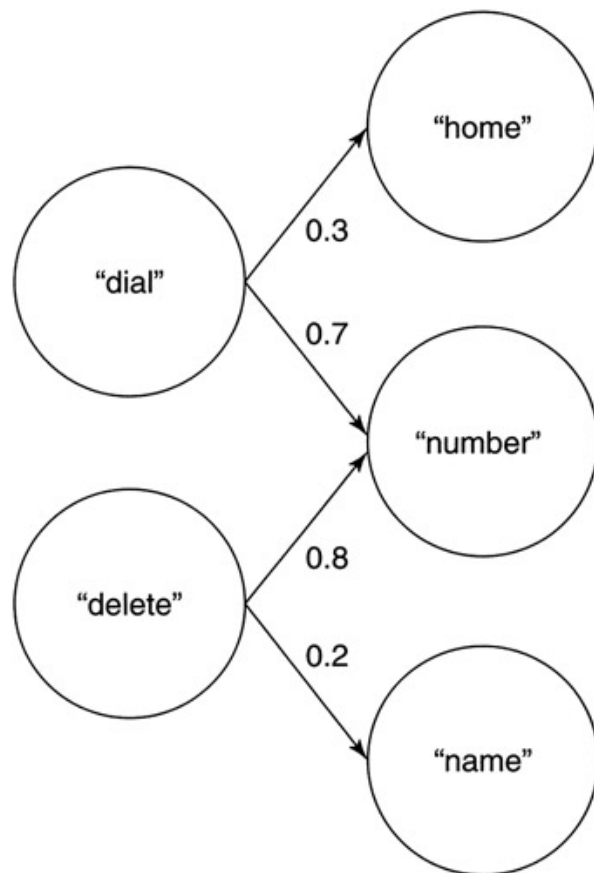
Let's now look at a few applications of Markov Chains for HMMs. The first example, speech recognition, is a practical method used to determine the words spoken in natural language processing. The following two examples are more thought provoking than useful, but provide a greater understanding of the possibilities with Markov Chains.

### Speech Recognition

Recall from [Figure 10.1](#) that the pronunciation of a word can take on one or more variations based upon the dialect or origin of the speaker. Building speech recognition systems then becomes very difficult, because the system must deal with a variety of pronunciations of a given word.

Hidden Markov Chains provide the opportunity to simplify speech recognition systems through probabilistic parsing of the phonemes of speech. For example, let's say that a speech system is designed to understand a handful of words, two of which are "tomorrow" and "today." When the system first hears the "tah" phoneme, the spoken word can be either "tomorrow" or "today." The next phoneme parsed from the speech is "m"; the probability that the word being spoken is "today" is now 0. Given our HMM, the "m" phoneme is a legal state, so we transition and then process the next phoneme. Given the probabilities at the transitions, the speech recognizer can use these to take the most likely path through the chain to identify the most likely phoneme that followed.

This example can be extended from phonemes to words. Given a set of words that can be understood by a speech recognizer, a Markov Chain can be created that identifies the probabilities of a given word following another. This would allow the recognizer to better identify which words were spoken, based upon their context (see [Figure 10.3](#)).



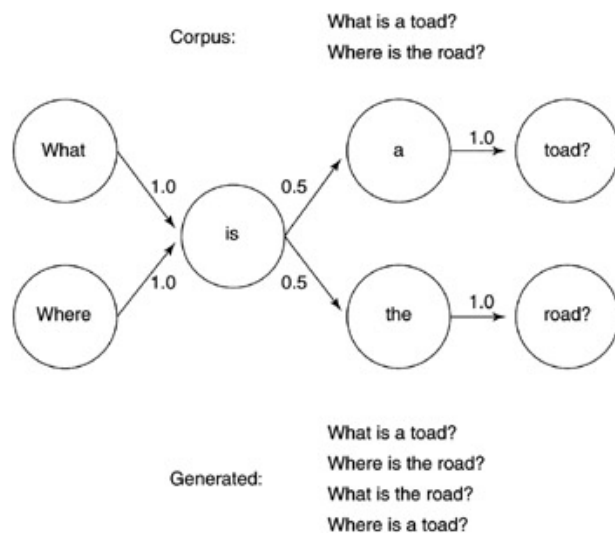
**Figure 10.3:** Higher level speech recognizer Markov Chain.

It's clear from these examples that HMMs can greatly simplify tasks such as speech recognition and speech understanding. In this example, the phoneme inputs caused the transition of states within the model to define words. Further, word inputs caused a transition for contextual understanding of a sentence. Next, we'll look at the use of HMMs to generate symbols based upon predefined or learned transition probabilities.

### Modeling Text

In the previous examples, the Markov Chain was used to probabilistically identify the next state given the current state and an external stimulus. Let's now look at some examples where no external stimuli is provided—transitions between the states of the Markov Chain are based solely on the defined probabilities as a random process.

In [Figure 10.4](#), an example Markov Chain is shown for two sample sentences. The Markov Chain is a product of these two sentences, using the bigram model.



**Figure 10.4:** Sample bigram from a corpus of seven unique words.

The only non-unique word within this corpus is the word "is." With even probability, the word "is" can lead either to the word "a" or "the." Note that this now leads to four possible sentences that can be generated with the Markov Chain (shown at the bottom of [Figure 10.4](#)).

## Modeling Music

In a similar fashion to words, consider training the HMM from a vocabulary of musical notes from a given composer [[Zhang 2001](#)]. The HMM could then be used to compose a symphony via the probabilistic generation of notes with a style of the given composer. Consider also the training of an HMM from a vocabulary of two or more composers. With a large enough n-gram, symphonies could be arranged from the combinations of great composers.

## Sample Application

Some suggest, using higher approximations of the HMM, that it might be possible to mimic the works of great writers such as Shakespeare [Zhang 2001]. By training the HMM with a corpus of the author's text, the HMM could then be used to emit sequences of words that are statistically similar to those from the training corpus.

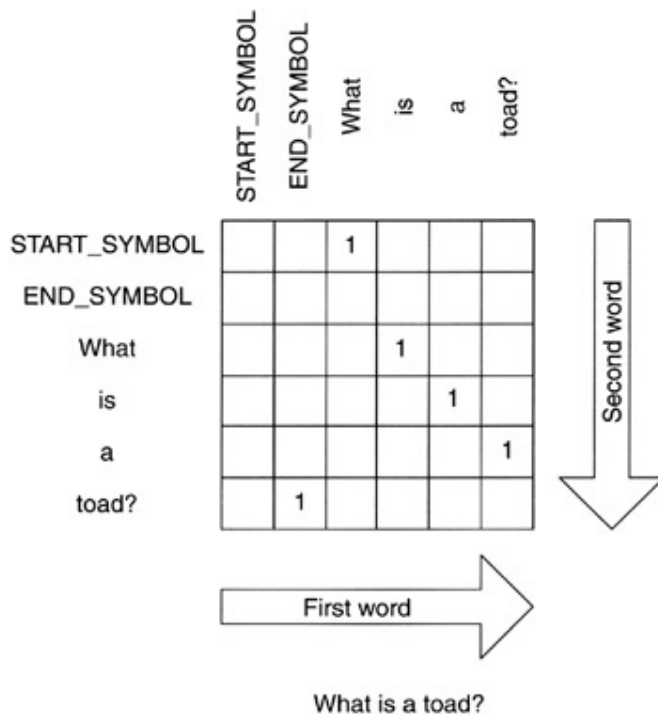
For the sample application, we'll discuss the implementation of a bigram HMM that can be trained with arbitrary text. The HMM will then be used to generate random sequences of characters.

**On the CD** The source code for the bigram HMM can be found on the CD-ROM at `./software/ch10/bigram/`.



the sentence, we update the START\_SYMBOL row (the first row in the table), and increment the column offset defined by the wordIndex. We also update the sum or the row as defined by the sumVector array (used to calculate the relative probabilities for each word).

If the order of the current word is the last word, then we update the bigramArray using the word as the first dimension of the array and the LAST\_SYMBOL as the second dimension. Finally, if the word is in the middle, we use the last word as the first index and the current word as the second index. The last word is always saved within the function within the lastIndex variable. See [Figure 10.5](#) for a representation of a sample sentence.



**Figure 10.5:** bigramArray for the sample sentence.

This completes the parsing and filling in of the bigram array. The next two functions provide for emitting a sentence using the bigram array as the model.

Function buildSentence walks through the bigramArray structure using the sumVector array to determine which path to take and thus

## Samples

Let's now look at some samples of the algorithm using a variety of different text inputs.

In the first example, a set of quotes from Albert Einstein was used as the training file. This training set consists of 13 quotes with 377 unique words. See [Listing 10.9](#) for example products from the Einstein corpus.

### **Listing 10.9: Emitted Text from Albert Einstein Quotes.**



```
[root@plato bigram] ./bigram -f einstein
```

```
Imagination is shipwrecked by language and other  
symbolic devices.
```

```
[root@plato bigram] ./bigram -f einstein
```

```
We all ruled in the authority of imagination.
```



Each of the generated quotes from [Listing 10.9](#) sound reasonable and is actually thought provoking.

Consider now generated text from a corpus of Albert Camus' *Absurd Man* (see [Listing 10.10](#)).

### **Listing 10.10: Emitted Text from Albert Camus' *Absurd Man*.**




```
[root@plato bigram] ./bigram -f absurdman
```

```
It is to speak only truth that is the breath of
```

future actions.

```
[root@plato bigram] ./bigram -f absurdman  
"My field" said Goethe "is time and unfolds in a  
disservice to  
make.
```




Finally, consider a Markov Chain that was created from two separate works (see [Listing 10.11](#)). The first work is the poem "Ode to Joy" by anonymous, and the second from book one of *The Odyssey* by Homer.

#### **Listing 10.11: Emitted Text from a Collection of Works.**



```
[root@plato bigram] ./bigram -f combo
```

```
From chaos and beasts and hostile shores!  From  
the woods the  
hunter strayed.
```



## Ownership?

While the Markov Chain could be used to mimic a symphony, or the generation of text that appears similar to other literature, the question arises as to who owns the new work? The Markov Chain can emit music or text that is modeled after the original training data. Therefore, while the result is very similar to the original author's work, it is a new statistical representation of the original work.

When trigram or larger models are used to model the original work, the complexity and coherence of the resulting work will be extremely interesting.

## Summary

In this chapter, we investigated Markov Chains and the bigram model for text generation. Markov Chains can be used in a variety of applications, from spelling checkers to the confirmation of the authorship of an unknown work. The applications of speech recognition and text and music modeling were discussed as well as an implementation of a bigram text generator that emits interesting phrases. Finally, the issue of legal ownership was touched upon since emitted works of Markov Chains are based solely on the statistical representations of another author's work.

## References

[Henke 1999] Henke, Jonathon. (1999). "Statistical Inference: n-gram Models over Sparse Data", TDM Seminar. Available online at <http://www.sims.berkeley.edu/courses/is296a-4/f99/Lectures/henke-ch6.ppt>(accessed January 17, 2003)

[Zheng 2001] Zheng, Byoung-Tak. (2001). "Course 4190.515: Bioinformatics (Machine Learning)," Seoul National University School of Computer Science and Engineering. Available online at <http://bi.snu.ac.kr/Courses/g-ai01/g-ai01.html>(accessed January 17, 2003)

## Resources

Baker, J.K. (1975). "Stochastic Modeling for Automatic Speech Understanding." In R. Reddy (ed.), *Speech Recognition* ( pp. 521–42). Academic Press.

Baldi, P., and S. Brunak. (1998). *Bioinformatics: The Machine Learning Approach*. Cambridge, Mass.: MIT Press.

Shannon, C. E. (1948). "A mathematical theory of communication," *Bell System Technical Journal* 27(July and October): 379–423 and 623–56.

Shannon, C. E. (1951). "Prediction and Entropy of Printed English," *Bell System Technical Journal* 30:50–64.