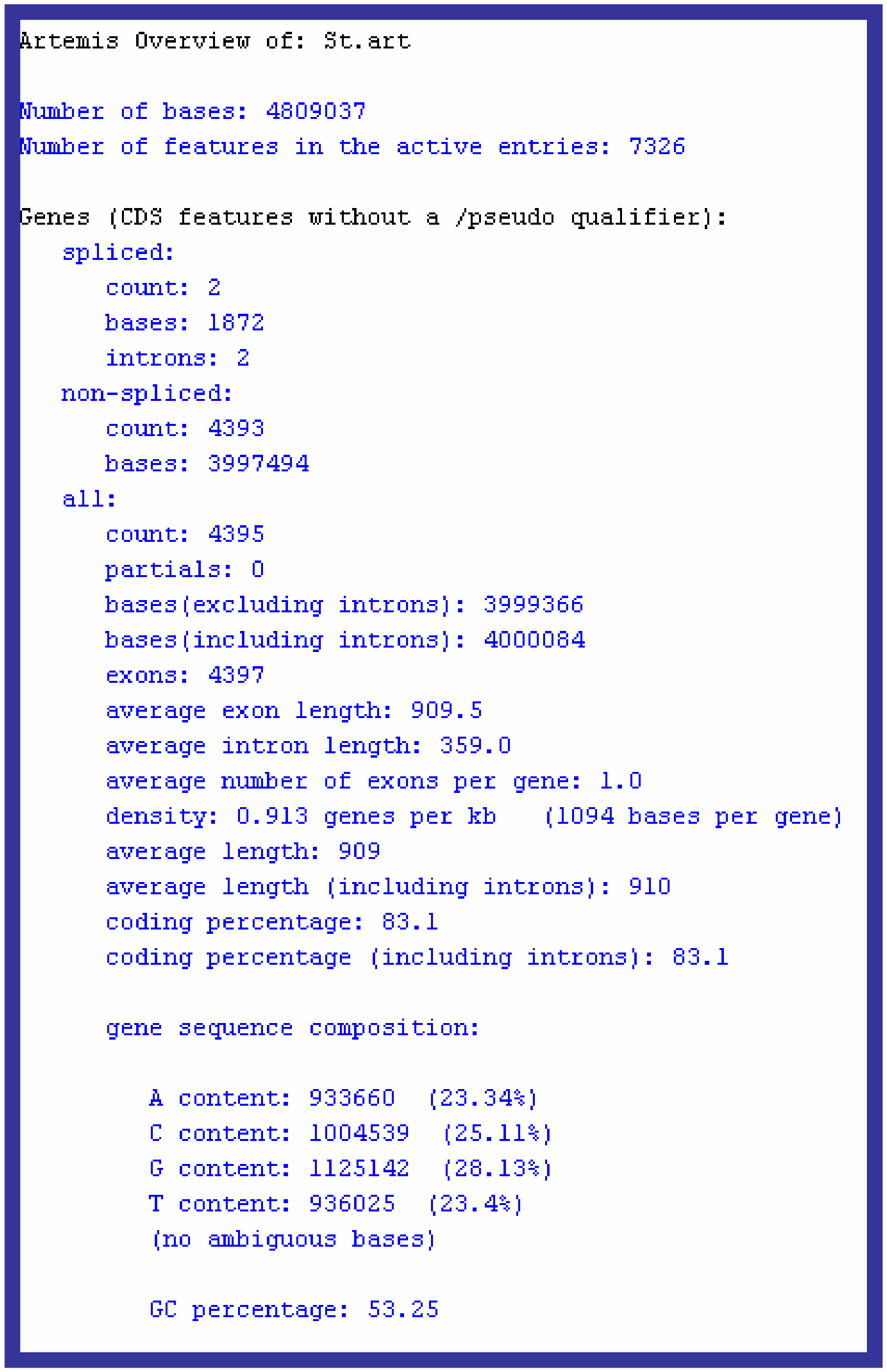
Genome Atlas

http://www.cbs.dtu.dk/services/GenomeAtlas/



```
Artemis Overview of: St.art

Number of bases: 4809037
Number of features in the active entries: 7326

Genes (CDS features without a /pseudo qualifier):
   spliced:
      count: 2
      bases: 1872
      introns: 2
   non-spliced:
      count: 4393
      bases: 3997494
   all:
      count: 4395
      partials: 0
      bases(excluding introns): 3999366
      bases(including introns): 4000084
      exons: 4397
      average exon length: 909.5
      average intron length: 359.0
      average number of exons per gene: 1.0
      density: 0.913 genes per kb   (1094 bases per gene)
      average length: 909
      average length (including introns): 910
      coding percentage: 83.1
      coding percentage (including introns): 83.1

      gene sequence composition:

         A content: 933660  (23.34%)
         C content: 1004539  (25.11%)
         G content: 1125142  (28.13%)
         T content: 936025  (23.4%)
         (no ambiguous bases)

         GC percentage: 53.25
```

544–548 *Nucleic Acids Research, 1998, Vol. 26, No. 2* © 1998 Oxford University Press

Microbial gene identification using interpolated Markov models

Steven L. Salzberg[1,2,*], Arthur L. Delcher[3], Simon Kasif[4] and Owen White[1]

[1]The Institute for Genomic Research, 9712 Medical Center Drive, Rockville, MD 20850, USA, [2]Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA, [3]Department of Computer Science, Loyola College in Maryland, Baltimore, MD 21210, USA and [4]Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA

Received September 10, 1997; Revised and Accepted November 11, 1997

ABSTRACT

This paper describes a new system, GLIMMER, for finding genes in microbial genomes. In a series of tests on *Haemophilus influenzae*, *Helicobacter pylori* and other complete microbial genomes, this system has proven to be very accurate at locating virtually all the genes in these sequences, outperforming previous methods. A conservative estimate based on experiments on *H.pylori* and *H.influenzae* is that the system finds >97% of all genes. GLIMMER uses interpolated Markov models (IMMs) as a framework for capturing dependencies between nearby nucleotides in a DNA sequence. An IMM-based method makes predictions based on a variable context; i.e., a variable-length oligomer in a DNA sequence. The context used by GLIMMER changes depending on the local composition of the sequence. As a result, GLIMMER is more flexible and more powerful than fixed-order Markov methods, which have previously been the primary content-based technique for finding genes in microbial DNA.

INTRODUCTION

The number of new microbial genomes has dramatically increased since the first genome, *Haemophilus influenzae*, was sequenced in 1995 (1). Ten whole genomes have been completed, and at least 30 others are expected to be completed in the next two years. This abundance of data demands new and highly accurate computational analysis tools in order to explore these genomes and maximize the scientific knowledge gained from them. One of the first steps in the analysis of a microbial genome is the identification of all its genes. Because these genomes tend to be gene-rich, typically containing 90% coding sequence, the gene discovery problem takes on a in new genomes still have no significant homology to known genes (1). For these genes, we must rely on computational methods of scoring the coding region to identify the genes. The best-known program for this task is GeneMark (5), which uses a Markov chain model to score coding regions. GeneMark has been highly effective and was used in the *H.influenza* and more recent genome projects. We have developed a new system, GLIMMER, that uses a technique called interpolated Markov models (IMMs) to find coding regions in microbial sequences. IMMs are in principle more powerful than Markov chains, and the computational experiments described below demonstrate that they produce more accurate results when used to find genes in bacterial DNA.

Markov models are a well-known tool for analyzing biological sequence data, and the predominant model for microbial sequence analysis is a fixed-order Markov chain (5,6). A fixed order Markov model predicts each base of a DNA sequence using a fixed number of preceding bases in the sequence. For example, a 5th-order model, which is the basis of GeneMark, uses the five previous bases to predict the next base. However, learning such models accurately can be difficult when there is insufficient training data to accurately estimate the probability of each base occurring after every possible combination of five preceding bases. In general, a kth-order Markov model for DNA sequences requires 4^{k+1} probabilities to be estimated from the training data (e.g., 4096 probabilities for a 5th-order model). In order to estimate these probabilities, many occurrences of all possible $k$mers must be present in the data.

An IMM overcomes this problem by combining probabilities from contexts of varying lengths to make predictions, and by only using those contexts (oligomers) for which sufficient data are available. In a typical microbial genome some 5mers will occur too infrequently to give reliable estimates of the probability of the next base, while some 8mers may occur frequently enough to give very reliable estimates. In principle, using longer oligomers is always preferable to using shorter ones, but only if sufficient data is

models (IMMs) as a framework for capturing dependencies between nearby nucleotides in a DNA sequence. An IMM-based method makes predictions based on a variable context; i.e., a variable-length oligomer in a DNA sequence. The context used by GLIMMER changes depending on the local composition of the sequence. As a result, GLIMMER is more flexible and more powerful than fixed-order Markov methods, which have previously been the primary content-based technique for finding genes in microbial DNA.

INTRODUCTION

The number of new microbial genomes has dramatically increased since the first genome, *Haemophilus influenzae*, was sequenced in 1995 (1). Ten whole genomes have been completed, and at least 30 others are expected to be completed in the next two years. This abundance of data demands new and highly accurate computational analysis tools in order to explore these genomes and maximize the scientific knowledge gained from them. One of the first steps in the analysis of a microbial genome is the identification of all its genes. Because these genomes tend to be gene-rich, typically containing 90% coding sequence, the gene discovery problem takes on a different character than it does in eukaryotic genomes, especially higher eukaryotes whose genomes may have <10% coding sequence. In particular, the most difficult problem is determining which of two or more overlapping open reading frames (orfs) represent true genes. Other difficult problems include identifying the start of translation and finding regulatory signals such as promoters and terminators.

The most reliable way to identify a gene in a new genome is to find a close homolog from another organism. This can be done today very effectively using programs such as BLAST (3) and FASTA (4) to search all the entries in GenBank. However, many of the genes

find genes in bacterial DNA.

Markov models are a well-known tool for analyzing biological sequence data, and the predominant model for microbial sequence analysis is a fixed-order Markov chain (5,6). A fixed order Markov model predicts each base of a DNA sequence using a fixed number of preceding bases in the sequence. For example, a 5th-order model, which is the basis of GeneMark, uses the five previous bases to predict the next base. However, learning such models accurately can be difficult when there is insufficient training data to accurately estimate the probability of each base occurring after every possible combination of five preceding bases. In general, a k^{th}-order Markov model for DNA sequences requires 4^{k+1} probabilities to be estimated from the training data (e.g., 4096 probabilities for a 5th-order model). In order to estimate these probabilities, many occurrences of all possible *k*mers must be present in the data.

An IMM overcomes this problem by combining probabilities from contexts of varying lengths to make predictions, and by only using those contexts (oligomers) for which sufficient data are available. In a typical microbial genome some 5mers will occur too infrequently to give reliable estimates of the probability of the next base, while some 8mers may occur frequently enough to give very reliable estimates. In principle, using longer oligomers is always preferable to using shorter ones, but only if sufficient data is available to produce good probability estimates. An IMM uses a linear combination of probabilities obtained from several lengths of oligomers to make predictions, giving high weights to oligomers that occur frequently and low weights to those that do not. Thus an IMM uses a longer context to make a prediction whenever possible, taking advantage of the greater accuracy produced by higher-order Markov models. Where the statistics on longer oligomers are insufficient to produce good estimates, an IMM can fall back on shorter oligomers to make its predictions.

Using IMMs we have developed a new system, called GLIMMER, to identify coding regions in microbial DNA.

ABSTRACT

This paper describes a new system, GLIMMER, for finding genes in microbial genomes. In a series of tests on *Haemophilus influenzae*, *Helicobacter pylori* and other complete microbial genomes, this system has proven to be very accurate at locating virtually all the genes in these sequences, outperforming previous methods. A conservative estimate based on experiments on *H.pylori* and *H.influenzae* is that the system finds >97% of all genes. GLIMMER uses interpolated Markov models (IMMs) as a framework for capturing dependencies between nearby nucleotides in a DNA sequence. An IMM-based method makes predictions based on a variable context; i.e., a variable-length oligomer in a DNA sequence. The context used by GLIMMER changes depending on the local composition of the sequence. As a result, GLIMMER is more flexible and more powerful than fixed-order Markov methods, which have previously been the primary content-based technique for finding genes in microbial DNA.

INTRODUCTION

The number of new microbial genomes has dramatically increased since the first genome, *Haemophilus influenzae*, was sequenced in 1995 (1). Ten whole genomes have been completed, and at least 30 others are expected to be completed in the next two years. This abundance of data demands new and highly accurate computational analysis tools in order to explore these genomes and maximize the scientific knowledge gained from them. One of the first steps in the in new genomes still have no significant homology to known genes (1). For these genes, we must rely on computational methods of scoring the coding region to identify the genes. The best-known program for this task is GeneMark (5), which uses a Markov chain model to score coding regions. GeneMark has been highly effective and was used in the *H.influenza* and more recent genome projects. We have developed a new system, GLIMMER, that uses a technique called interpolated Markov models (IMMs) to find coding regions in microbial sequences. IMMs are in principle more powerful than Markov chains, and the computational experiments described below demonstrate that they produce more accurate results when used to find genes in bacterial DNA.

Markov models are a well-known tool for analyzing biological sequence data, and the predominant model for microbial sequence analysis is a fixed-order Markov chain (5,6). A fixed order Markov model predicts each base of a DNA sequence using a fixed number of preceding bases in the sequence. For example, a 5th-order model, which is the basis of GeneMark, uses the five previous bases to predict the next base. However, learning such models accurately can be difficult when there is insufficient training data to accurately estimate the probability of each base occurring after every possible combination of five preceding bases. In general, a kth-order Markov model for DNA sequences requires 4^{k+1} probabilities to be estimated from the training data (e.g., 4096 probabilities for a 5th-order model). In order to estimate these probabilities, many occurrences of all possible $k$mers must be present in the data.

An IMM overcomes this problem by combining probabilities from contexts of varying lengths to make predictions, and by only using those contexts (oligomers) for which sufficient data are available. In a typical microbial genome some 5mers will occur too infrequently to give reliable estimates of the probability of the next

ABSTRACT

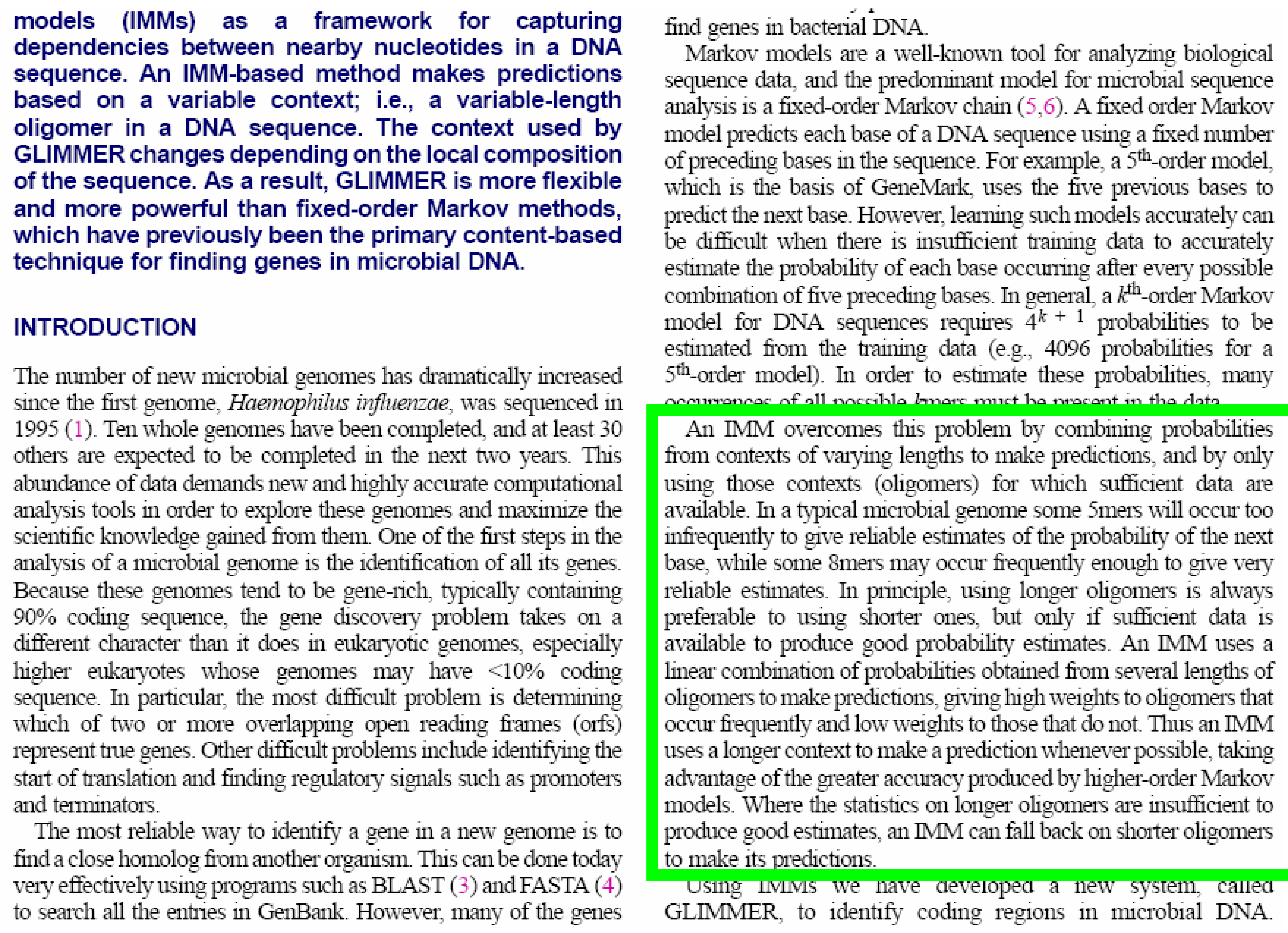
This paper describes a new system, GLIMMER, for finding genes in microbial genomes. In a series of tests on *Haemophilus influenzae*, *Helicobacter pylori* and other complete microbial genomes, this system has proven to be very accurate at locating virtually all the genes in these sequences, outperforming previous methods. A conservative estimate based on experiments on *H.pylori* and *H.influenzae* is that the system finds >97% of all genes. GLIMMER uses interpolated Markov models (IMMs) as a framework for capturing dependencies between nearby nucleotides in a DNA sequence. An IMM-based method makes predictions based on a variable context; i.e., a variable-length oligomer in a DNA sequence. The context used by GLIMMER changes depending on the local composition of the sequence. As a result, GLIMMER is more flexible and more powerful than fixed-order Markov methods, which have previously been the primary content-based technique for finding genes in microbial DNA.

INTRODUCTION

The number of new microbial genomes has dramatically increased since the first genome, *Haemophilus influenzae*, was sequenced in 1995 (1). Ten whole genomes have been completed, and at least 30 others are expected to be completed in the next two years. This abundance of data demands new and highly accurate computational analysis tools in order to explore these genomes and maximize the scientific knowledge gained from them. One of the first steps in the in new genomes still have no significant homology to known genes (1). For these genes, we must rely on computational methods of scoring the coding region to identify the genes. The best-known program for this task is GeneMark (5), which uses a Markov chain model to score coding regions. GeneMark has been highly effective and was used in the *H.influenza* and more recent genome projects. We have developed a new system, GLIMMER, that uses a technique called interpolated Markov models (IMMs) to find coding regions in microbial sequences. IMMs are in principle more powerful than Markov chains, and the computational experiments described below demonstrate that they produce more accurate results when used to find genes in bacterial DNA.

Markov models are a well-known tool for analyzing biological sequence data, and the predominant model for microbial sequence analysis is a fixed-order Markov chain (5,6). A fixed order Markov model predicts each base of a DNA sequence using a fixed number of preceding bases in the sequence. For example, a 5th-order model, which is the basis of GeneMark, uses the five previous bases to predict the next base. However, learning such models accurately can be difficult when there is insufficient training data to accurately estimate the probability of each base occurring after every possible combination of five preceding bases. In general, a kth-order Markov model for DNA sequences requires 4^{k+1} probabilities to be estimated from the training data (e.g., 4096 probabilities for a 5th-order model). In order to estimate these probabilities, many occurrences of all possible $k$mers must be present in the data.

An IMM overcomes this problem by combining probabilities from contexts of varying lengths to make predictions, and by only using those contexts (oligomers) for which sufficient data are available. In a typical microbial genome some 5mers will occur too infrequently to give reliable estimates of the probability of the next



models (IMMs) as a framework for capturing dependencies between nearby nucleotides in a DNA sequence. An IMM-based method makes predictions based on a variable context; i.e., a variable-length oligomer in a DNA sequence. The context used by GLIMMER changes depending on the local composition of the sequence. As a result, GLIMMER is more flexible and more powerful than fixed-order Markov methods, which have previously been the primary content-based technique for finding genes in microbial DNA.

INTRODUCTION

The number of new microbial genomes has dramatically increased since the first genome, *Haemophilus influenzae*, was sequenced in 1995 (1). Ten whole genomes have been completed, and at least 30 others are expected to be completed in the next two years. This abundance of data demands new and highly accurate computational analysis tools in order to explore these genomes and maximize the scientific knowledge gained from them. One of the first steps in the analysis of a microbial genome is the identification of all its genes. Because these genomes tend to be gene-rich, typically containing 90% coding sequence, the gene discovery problem takes on a different character than it does in eukaryotic genomes, especially higher eukaryotes whose genomes may have <10% coding sequence. In particular, the most difficult problem is determining which of two or more overlapping open reading frames (orfs) represent true genes. Other difficult problems include identifying the start of translation and finding regulatory signals such as promoters and terminators.

The most reliable way to identify a gene in a new genome is to find a close homolog from another organism. This can be done today very effectively using programs such as BLAST (3) and FASTA (4) to search all the entries in GenBank. However, many of the genes

find genes in bacterial DNA.

Markov models are a well-known tool for analyzing biological sequence data, and the predominant model for microbial sequence analysis is a fixed-order Markov chain (5,6). A fixed order Markov model predicts each base of a DNA sequence using a fixed number of preceding bases in the sequence. For example, a 5th-order model, which is the basis of GeneMark, uses the five previous bases to predict the next base. However, learning such models accurately can be difficult when there is insufficient training data to accurately estimate the probability of each base occurring after every possible combination of five preceding bases. In general, a kth-order Markov model for DNA sequences requires 4^{k+1} probabilities to be estimated from the training data (e.g., 4096 probabilities for a 5th-order model). In order to estimate these probabilities, many occurrences of all possible *k*mers must be present in the data.

An IMM overcomes this problem by combining probabilities from contexts of varying lengths to make predictions, and by only using those contexts (oligomers) for which sufficient data are available. In a typical microbial genome some 5mers will occur too infrequently to give reliable estimates of the probability of the next base, while some 8mers may occur frequently enough to give very reliable estimates. In principle, using longer oligomers is always preferable to using shorter ones, but only if sufficient data is available to produce good probability estimates. An IMM uses a linear combination of probabilities obtained from several lengths of oligomers to make predictions, giving high weights to oligomers that occur frequently and low weights to those that do not. Thus an IMM uses a longer context to make a prediction whenever possible, taking advantage of the greater accuracy produced by higher-order Markov models. Where the statistics on longer oligomers are insufficient to produce good estimates, an IMM can fall back on shorter oligomers to make its predictions.

Using IMMs we have developed a new system, called GLIMMER, to identify coding regions in microbial DNA.

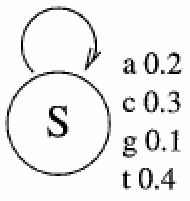


Figure 1. Sample 1-state Markov model for simple sequence modeling.

GLIMMER uses a novel approach, based on frequency of occurrence and predictive value, to determine the relative weights of oligomers that vary in length from 1 to 8. After first creating IMMs for each of the six possible reading frames, GLIMMER then uses them to score entire orfs. When two high-scoring orfs overlap, the overlap region is scored separately to determine which orf is more likely to be a gene. We have tested GLIMMER using the *H.influenzae*, *Helicobacter pylori* and *Escherichia coli* genomes and found that it is very accurate in identifying genes, as we explain in Methods and Results. The system has recently been used to find the genes in two newly completed genomes: *Borrelia burgdorferi*, the bacteria that causes Lyme disease (14), and *Treponema pallidum*, the bacteria that causes syphilis (Fraser *et al.*, manuscript in preparation). Annotation for these and other completed genomes will be available on the GLIMMER web site.

INTERPOLATED MARKOV MODELS

Markov chains

Our probabilistic model of DNA sequences represents a sequence as a process that may be described as a sequence of random variables $X_1, X_2, ...,$ where X_i corresponds to position i in the sequence. Each random variable X_i takes a value from the set of bases (a, c, g, t). The probability that a variable X_i takes will depend on the local context; that is, the bases immediately adjacent to the base at position i. We

bases in the three codon positions. Even with a 0th-order model, the frequency of g in codon position 1 will be different from its frequency in another frame, so even this very weak model has some ability to identify the right reading frame for a gene.

In a 1st-order model, the output of a state depends on the state immediately previous; i.e., a base is dependent on the previous base. Thus instead of four probabilities in each state, we compute sixteen: $p(a|a)$, $p(a|c)$, ..., $p(t|t)$. In order to score a new sequence, the model considers two bases at a time, the current base and the previous one. Likewise, in a 2nd-order model, the output of a state depends on the two previous bases. So to predict a base in the third codon position with our 2nd-order model, we look at the first and second codon positions. To predict a base in the first codon position, the 2nd-order model looks at the second and third codon positions in the previous codon.

Using the Markov models for each of the six possible frames plus a model of non-coding DNA, we can straightforwardly produce a simple algorithm for finding genes. Simply score every orf using all seven models, and choose the model with the highest score. The scores can be normalized so they represent the probability that a sequence is coding. If the model corresponding to the true coding region in the correct frame scores the highest, then the orf can be labeled as a gene. This simple algorithm ignores the difficult problem of how to handle overlapping genes, which we address in the Algorithm and System Design section, which contains the details of GLIMMER. (To be effective, an algorithm must do much more than this intentionally simple description. For example, all scores could be nearly equal, or the highest score could still be quite low, so the algorithm needs to have a threshold score below which no region is classified as coding.)

Interpolated models

INTERPOLATED MARKOV MODELS

Markov chains

Our probabilistic model of DNA sequences represents a sequence as a process that may be described as a sequence of random variables $X_1, X_2, ...,$ where X_i corresponds to position i in the sequence. Each random variable X_i takes a value from the set of bases (a, c, g, t). The probability that a variable X_i takes will depend on the local context; that is, the bases immediately adjacent to the base at position i. We sometimes refer to (a, c, g, t) as the set of possible *states* that a variable can take. In other words, variable X_i is in state a if $X_i = a$. As an illustration, consider the simple example of a Markov model in Figure 1. This 1-state model can be used to model any length DNA sequence. In each position, the probability of a is 0.2. Thus the sequence aaaaa would have a probability of $(0.2)^5 = 0.00032$. In this way we can score any sequence by computing the probability that it was generated by the model.

A first order Markov chain is a sequence of random variables where the probability that X_i takes a particular value only depends on the preceding variable X_{i-1}. A k^{th} order Markov chain is a natural generalization of this definition where the probability distribution of X_i depends only on the k preceding bases. Note that for DNA sequences a first-order Markov chain is specified completely by a matrix of 16 probabilities: $p(a|a), p(a|c), ..., p(t|t)$. There are two essential computational issues that must be considered in building and using these probabilistic models: (i) the learning problem, which involves learning a good model for coding regions in microbial DNA and (ii) the evaluation problem, which involves assigning a score to a new DNA sequence that represents the likelihood that the sequence is coding. GLIMMER's solutions to both these computational issues are described in the Interpolated models section below.

problem of how to handle overlapping genes, which we address in the Algorithm and System Design section, which contains the details of GLIMMER. (To be effective, an algorithm must do much more than this intentionally simple description. For example, all scores could be nearly equal, or the highest score could still be quite low, so the algorithm needs to have a threshold score below which no region is classified as coding.)

Interpolated models

In general, we would always like to use the highest-order Markov model possible. The higher-order model should always do at least as well as, and frequently better than, lower-order models. This can be explained by a simple example.

Suppose that the base in the third codon position depends only on the second codon position. Then we might observe in a given genome that $P(a_3|g_2) = 0.22$; i.e., the probability of observing adenine in the third codon position given that guanine occurs in the second is 0.22. This is a first-order dependency. Suppose that the prior probability of adenine $P(a_3)$ is 0.30. Clearly we will perform better by using the first-order statistic, since adenine occurs less frequently in the third position following guanine than it does otherwise. Now consider using both the first and second codon positions to predict a_3. Given our assumption that only the second position matters, we should find that $P(a_3|g_2) = P(a_3|g_2, x_1)$, where x_1 indicates any base in the first codon position. Thus the 2nd-order model will perform exactly the same as the 1st-order model. If it turns out that the third codon position depends on both the first and second positions, then the 2nd-order model will perform better.

The problem that arises in practice is that, as we move to higher order models, the number of probabilities that we must estimate from the data increases exponentially. For DNA sequence data, we need to learn 4^{k+1} probabilities in a k^{th}-order Markov model. Our random variable X_i takes a value from the set of bases (a, c, g, t). The probability that a variable X_i takes will depend on the local context; that is, the bases immediately adjacent to the base at position i. We sometimes refer to (a, c, g, t) as the set of possible *states* that a variable can take. In other words, variable X_i is in state a if $X_i = a$. As an illustration, consider the simple example of a Markov model in Figure 1. This 1-state model can be used to model any length DNA sequence. In each position, the probability of a is 0.2. Thus the sequence aaaaa would have a probability of $(0.2)^5 = 0.00032$. In this way we can score any sequence by computing the probability that it was generated by the model.

A first order Markov chain is a sequence of random variables where the probability that X_i takes a particular value only depends on the preceding variable X_{i-1}. A k^{th} order Markov chain is a natural generalization of this definition where the probability distribution of X_i depends only on the k preceding bases. Note that for DNA sequences a first-order Markov chain is specified completely by a matrix of 16 probabilities: $p(a|a), p(a|c), ..., p(t|t)$. There are two essential computational issues that must be considered in building and using these probabilistic models: (i) the learning problem, which involves learning a good model for coding regions in microbial DNA and (ii) the evaluation problem, which involves assigning a score to a new DNA sequence that represents the likelihood that the sequence is coding. GLIMMER's solutions to both these computational issues are described in the Interpolated models section below.

To use a Markov chain model to find genes in microbial DNA, we need to build at least six submodels, one for each of the possible reading frames (three forward and three reverse). We can also build a seventh, separate model for non-coding regions, though this is not strictly necessary. Each model makes different predictions for the

Interpolated models

In general, we would always like to use the highest-order Markov model possible. The higher-order model should always do at least as well as, and frequently better than, lower-order models. This can be explained by a simple example.

Suppose that the base in the third codon position depends only on the second codon position. Then we might observe in a given genome that $P(a_3|g_2) = 0.22$; i.e., the probability of observing adenine in the third codon position given that guanine occurs in the second is 0.22. This is a first-order dependency. Suppose that the prior probability of adenine $P(a_3)$ is 0.30. Clearly we will perform better by using the first-order statistic, since adenine occurs less frequently in the third position following guanine than it does otherwise. Now consider using both the first and second codon positions to predict a_3. Given our assumption that only the second position matters, we should find that $P(a_3|g_2) = P(a_3|g_2, x_1)$, where x_1 indicates any base in the first codon position. Thus the 2nd-order model will perform exactly the same as the 1st-order model. If it turns out that the third codon position depends on both the first and second positions, then the 2nd-order model will perform better.

The problem that arises in practice is that, as we move to higher order models, the number of probabilities that we must estimate from the data increases exponentially. For DNA sequence data, we need to learn 4^{k+1} probabilities in a k^{th}-order Markov model. Our six submodels actually need $6 \times 4^{k+1}$ probabilities. So a 5th-order model needs 24 576 probabilities. In a microbial genome such as *H.influenzae* with 1.8 million bases, we will observe each of the 4096 possible 6mers often enough to get accurate estimates for a 5th-order model, although for rare hexamers we may not have

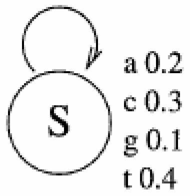


Figure 1. Sample 1-state Markov model for simple sequence modeling.

GLIMMER uses a novel approach, based on frequency of occurrence and predictive value, to determine the relative weights of oligomers that vary in length from 1 to 8. After first creating IMMs for each of the six possible reading frames, GLIMMER then uses them to score entire orfs. When two high-scoring orfs overlap, the overlap region is scored separately to determine which orf is more likely to be a gene. We have tested GLIMMER using the *H.influenzae*, *Helicobacter pylori* and *Escherichia coli* genomes and found that it is very accurate in identifying genes, as we explain in Methods and Results. The system has recently been used to find the genes in two newly completed genomes: *Borrelia burgdorferi*, the bacteria that causes Lyme disease (14), and *Treponema pallidum*, the bacteria that causes syphilis (Fraser *et al*., manuscript in preparation). Annotation for these and other completed genomes will be available on the GLIMMER web site.

INTERPOLATED MARKOV MODELS

Markov chains

Our probabilistic model of DNA sequences represents a sequence as a process that may be described as a sequence of random variables $X_1, X_2, ...,$ where X_i corresponds to position i in the sequence. Each random variable X_i takes a value from the set of bases (a, c, g, t). The bases in the three codon positions. Even with a 0th-order model, the frequency of g in codon position 1 will be different from its frequency in another frame, so even this very weak model has some ability to identify the right reading frame for a gene.

In a 1st-order model, the output of a state depends on the state immediately previous; i.e., a base is dependent on the previous base. Thus instead of four probabilities in each state, we compute sixteen: $p(a|a), p(a|c), ..., p(t|t)$. In order to score a new sequence, the model considers two bases at a time, the current base and the previous one. Likewise, in a 2nd-order model, the output of a state depends on the two previous bases. So to predict a base in the third codon position with our 2nd-order model, we look at the first and second codon positions. To predict a base in the first codon position, the 2nd-order model looks at the second and third codon positions in the previous codon.

Using the Markov models for each of the six possible frames plus a model of non-coding DNA, we can straightforwardly produce a simple algorithm for finding genes. Simply score every orf using all seven models, and choose the model with the highest score. The scores can be normalized so they represent the probability that a sequence is coding. If the model corresponding to the true coding region in the correct frame scores the highest, then the orf can be labeled as a gene. This simple algorithm ignores the difficult problem of how to handle overlapping genes, which we address in the Algorithm and System Design section, which contains the details of GLIMMER. (To be effective, an algorithm must do much more than this intentionally simple description. For example, all scores could be nearly equal, or the highest score could still be quite low, so the algorithm needs to have a threshold score below which no region is classified as coding.)

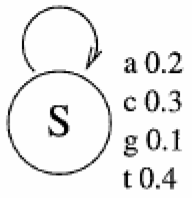


Figure 1. Sample 1-state Markov model for simple sequence modeling.

GLIMMER uses a novel approach, based on frequency of occurrence and predictive value, to determine the relative weights of oligomers that vary in length from 1 to 8. After first creating IMMs for each of the six possible reading frames, GLIMMER then uses them to score entire orfs. When two high-scoring orfs overlap, the overlap region is scored separately to determine which orf is more likely to be a gene. We have tested GLIMMER using the *H.influenzae*, *Helicobacter pylori* and *Escherichia coli* genomes and found that it is very accurate in identifying genes, as we explain in Methods and Results. The system has recently been used to find the genes in two newly completed genomes: *Borrelia burgdorferi*, the bacteria that causes Lyme disease (14), and *Treponema pallidum*, the bacteria that causes syphilis (Fraser *et al.*, manuscript in preparation). Annotation for these and other completed genomes will be available on the GLIMMER web site.

INTERPOLATED MARKOV MODELS

Markov chains

Our probabilistic model of DNA sequences represents a sequence as a process that may be described as a sequence of random variables $X_1, X_2, \ldots$, where X_i corresponds to position i in the sequence. Each random variable X_i takes a value from the set of bases (a, c, g, t). The

bases in the three codon positions. Even with a 0th-order model, the frequency of g in codon position 1 will be different from its frequency in another frame, so even this very weak model has some ability to identify the right reading frame for a gene.

In a 1st-order model, the output of a state depends on the state immediately previous; i.e., a base is dependent on the previous base. Thus instead of four probabilities in each state, we compute sixteen: $p(a|a)$, $p(a|c)$, ..., $p(t|t)$. In order to score a new sequence, the model considers two bases at a time, the current base and the previous one. Likewise, in a 2nd-order model, the output of a state depends on the two previous bases. So to predict a base in the third codon position with our 2nd-order model, we look at the first and second codon positions. To predict a base in the first codon position, the 2nd-order model looks at the second and third codon positions in the previous codon.

Using the Markov models for each of the six possible frames plus a model of non-coding DNA, we can straightforwardly produce a simple algorithm for finding genes. Simply score every orf using all seven models, and choose the model with the highest score. The scores can be normalized so they represent the probability that a sequence is coding. If the model corresponding to the true coding region in the correct frame scores the highest, then the orf can be labeled as a gene. This simple algorithm ignores the difficult problem of how to handle overlapping genes, which we address in the Algorithm and System Design section, which contains the details of GLIMMER. (To be effective, an algorithm must do much more than this intentionally simple description. For example, all scores could be nearly equal, or the highest score could still be quite low, so the algorithm needs to have a threshold score below which no region is classified as coding.)

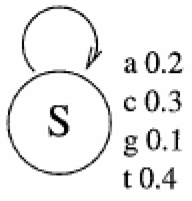


Figure 1. Sample 1-state Markov model for simple sequence modeling.

GLIMMER uses a novel approach, based on frequency of occurrence and predictive value, to determine the relative weights of oligomers that vary in length from 1 to 8. After first creating IMMs for each of the six possible reading frames, GLIMMER then uses them to score entire orfs. When two high-scoring orfs overlap, the overlap region is scored separately to determine which orf is more likely to be a gene. We have tested GLIMMER using the *H.influenzae*, *Helicobacter pylori* and *Escherichia coli* genomes and found that it is very accurate in identifying genes, as we explain in Methods and Results. The system has recently been used to find the genes in two newly completed genomes: *Borrelia burgdorferi*, the bacteria that causes Lyme disease (14), and *Treponema pallidum*, the bacteria that causes syphilis (Fraser *et al.*, manuscript in preparation). Annotation for these and other completed genomes will be available on the GLIMMER web site.

INTERPOLATED MARKOV MODELS

Markov chains

Our probabilistic model of DNA sequences represents a sequence as a process that may be described as a sequence of random variables $X_1, X_2, ...,$ where X_i corresponds to position i in the sequence. Each random variable X_i takes a value from the set of bases (*a, c, g, t*). The bases in the three codon positions. Even with a 0th-order model, the frequency of g in codon position 1 will be different from its frequency in another frame, so even this very weak model has some ability to identify the right reading frame for a gene.

In a 1st-order model, the output of a state depends on the state immediately previous; i.e., a base is dependent on the previous base. Thus instead of four probabilities in each state, we compute sixteen: $p(a|a)$, $p(a|c)$, ..., $p(t|t)$. In order to score a new sequence, the model considers two bases at a time, the current base and the previous one. Likewise, in a 2nd-order model, the output of a state depends on the two previous bases. So to predict a base in the third codon position with our 2nd-order model, we look at the first and second codon positions. To predict a base in the first codon position, the 2nd-order model looks at the second and third codon positions in the previous codon.

Using the Markov models for each of the six possible frames plus a model of non-coding DNA, we can straightforwardly produce a simple algorithm for finding genes. Simply score every orf using all seven models, and choose the model with the highest score. The scores can be normalized so they represent the probability that a sequence is coding. If the model corresponding to the true coding region in the correct frame scores the highest, then the orf can be labeled as a gene. This simple algorithm ignores the difficult problem of how to handle overlapping genes, which we address in the Algorithm and System Design section, which contains the details of GLIMMER. (To be effective, an algorithm must do much more than this intentionally simple description. For example, all scores could be nearly equal, or the highest score could still be quite low, so the algorithm needs to have a threshold score below which no region is classified as coding.)

probability that a variable X_i takes will depend on the local context; that is, the bases immediately adjacent to the base at position *i*. We sometimes refer to (*a, c, g, t*) as the set of possible *states* that a variable can take. In other words, variable X_i is in state *a* if $X_i = a$. As an illustration, consider the simple example of a Markov model in Figure 1. This 1-state model can be used to model any length DNA sequence. In each position, the probability of a is 0.2. Thus the sequence aaaaa would have a probability of $(0.2)^5 = 0.00032$. In this way we can score any sequence by computing the probability that it was generated by the model.

A first order Markov chain is a sequence of random variables where the probability that X_i takes a particular value only depends on the preceding variable X_{i-1}. A k^{th} order Markov chain is a natural generalization of this definition where the probability distribution of X_i depends only on the *k* preceding bases. Note that for DNA sequences a first-order Markov chain is specified completely by a matrix of 16 probabilities: $p(a|a), p(a|c), ..., p(t|t)$. There are two essential computational issues that must be considered in building and using these probabilistic models: (i) the learning problem, which involves learning a good model for coding regions in microbial DNA and (ii) the evaluation problem, which involves assigning a score to a new DNA sequence that represents the likelihood that the sequence is coding. GLIMMER's solutions to both these computational issues are described in the Interpolated models section below.

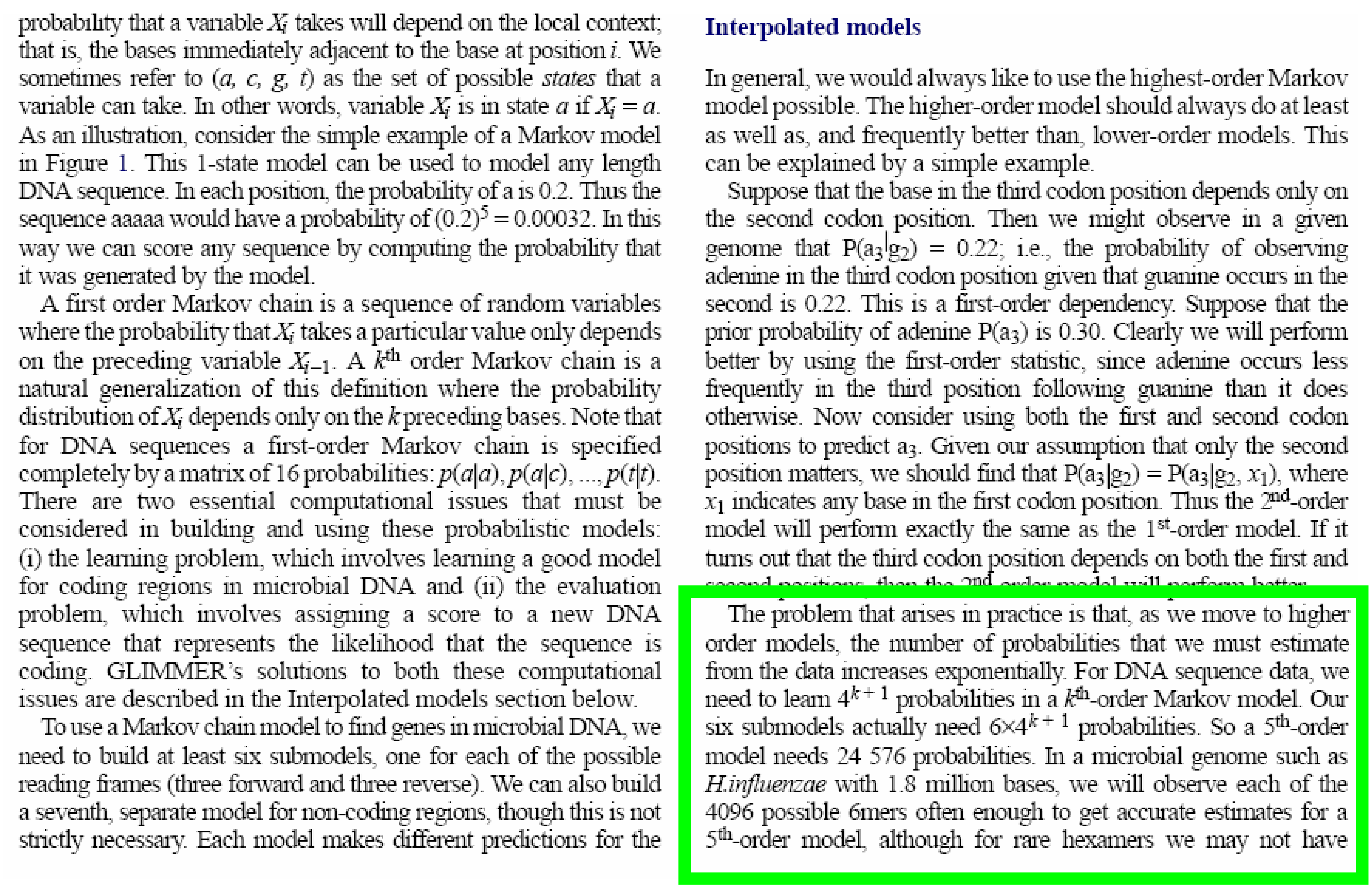
To use a Markov chain model to find genes in microbial DNA, we need to build at least six submodels, one for each of the possible reading frames (three forward and three reverse). We can also build a seventh, separate model for non-coding regions, though this is not strictly necessary. Each model makes different predictions for the

Interpolated models

In general, we would always like to use the highest-order Markov model possible. The higher-order model should always do at least as well as, and frequently better than, lower-order models. This can be explained by a simple example.

Suppose that the base in the third codon position depends only on the second codon position. Then we might observe in a given genome that $P(a_3|g_2) = 0.22$; i.e., the probability of observing adenine in the third codon position given that guanine occurs in the second is 0.22. This is a first-order dependency. Suppose that the prior probability of adenine $P(a_3)$ is 0.30. Clearly we will perform better by using the first-order statistic, since adenine occurs less frequently in the third position following guanine than it does otherwise. Now consider using both the first and second codon positions to predict a_3. Given our assumption that only the second position matters, we should find that $P(a_3|g_2) = P(a_3|g_2, x_1)$, where x_1 indicates any base in the first codon position. Thus the 2nd-order model will perform exactly the same as the 1st-order model. If it turns out that the third codon position depends on both the first and second positions, then the 2nd-order model will perform better.

The problem that arises in practice is that, as we move to higher order models, the number of probabilities that we must estimate from the data increases exponentially. For DNA sequence data, we need to learn 4^{k+1} probabilities in a k^{th}-order Markov model. Our six submodels actually need $6 \times 4^{k+1}$ probabilities. So a 5th-order model needs 24 576 probabilities. In a microbial genome such as *H.influenzae* with 1.8 million bases, we will observe each of the 4096 possible 6mers often enough to get accurate estimates for a 5th-order model, although for rare hexamers we may not have



probability that a variable X_i takes will depend on the local context; that is, the bases immediately adjacent to the base at position *i*. We sometimes refer to (*a, c, g, t*) as the set of possible *states* that a variable can take. In other words, variable X_i is in state *a* if $X_i = a$. As an illustration, consider the simple example of a Markov model in Figure 1. This 1-state model can be used to model any length DNA sequence. In each position, the probability of a is 0.2. Thus the sequence aaaaa would have a probability of $(0.2)^5 = 0.00032$. In this way we can score any sequence by computing the probability that it was generated by the model.

A first order Markov chain is a sequence of random variables where the probability that X_i takes a particular value only depends on the preceding variable X_{i-1}. A k^{th} order Markov chain is a natural generalization of this definition where the probability distribution of X_i depends only on the *k* preceding bases. Note that for DNA sequences a first-order Markov chain is specified completely by a matrix of 16 probabilities: $p(a|a), p(a|c), ..., p(t|t)$. There are two essential computational issues that must be considered in building and using these probabilistic models: (i) the learning problem, which involves learning a good model for coding regions in microbial DNA and (ii) the evaluation problem, which involves assigning a score to a new DNA sequence that represents the likelihood that the sequence is coding. GLIMMER's solutions to both these computational issues are described in the Interpolated models section below.

To use a Markov chain model to find genes in microbial DNA, we need to build at least six submodels, one for each of the possible reading frames (three forward and three reverse). We can also build a seventh, separate model for non-coding regions, though this is not strictly necessary. Each model makes different predictions for the

Interpolated models

In general, we would always like to use the highest-order Markov model possible. The higher-order model should always do at least as well as, and frequently better than, lower-order models. This can be explained by a simple example.

Suppose that the base in the third codon position depends only on the second codon position. Then we might observe in a given genome that $P(a_3|g_2) = 0.22$; i.e., the probability of observing adenine in the third codon position given that guanine occurs in the second is 0.22. This is a first-order dependency. Suppose that the prior probability of adenine $P(a_3)$ is 0.30. Clearly we will perform better by using the first-order statistic, since adenine occurs less frequently in the third position following guanine than it does otherwise. Now consider using both the first and second codon positions to predict a_3. Given our assumption that only the second position matters, we should find that $P(a_3|g_2) = P(a_3|g_2, x_1)$, where x_1 indicates any base in the first codon position. Thus the 2nd-order model will perform exactly the same as the 1st-order model. If it turns out that the third codon position depends on both the first and second positions, then the 2nd-order model will perform better.

The problem that arises in practice is that, as we move to higher order models, the number of probabilities that we must estimate from the data increases exponentially. For DNA sequence data, we need to learn 4^{k+1} probabilities in a k^{th}-order Markov model. Our six submodels actually need $6 \times 4^{k+1}$ probabilities. So a 5th-order model needs 24 576 probabilities. In a microbial genome such as *H.influenzae* with 1.8 million bases, we will observe each of the 4096 possible 6mers often enough to get accurate estimates for a 5th-order model, although for rare hexamers we may not have

enough data. For a 6th-order model, which requires probabilities for all 7mers, there are a substantial number of 7mers that do not occur sufficiently often, and for 7th and 8th-order models the problem is worse. However, even for 8th-order models, there are some oligomers that occur often enough to be extremely useful predictors. We would like a Markov model that uses these higher-order statistics whenever sufficient data is available. This is one of the key advantages of using an IMM. [Note that there exist other techniques to incorporate variable length predictive models (7,8). We experimented with these alternatives before converging on the approach described here.]

To be more precise, an IMM uses a combination of all the probabilities based on 0, 1, 2, ..., k previous bases, where k is a parameter given to the algorithm. In GLIMMER, we use $k = 8$. Thus for oligomers that occur frequently, the IMM can use an 8th-order model, while it might use a 5th or even lower-order model for rare oligomers. In order to 'smooth' its predictions, an IMM uses predictions from the lower-order models, where much more data is available, to adjust the predictions made from higher-order models.

During training, GLIMMER computes the probability of each base a, c, g, t, following all $k$mers for $0 \le k \le 8$. Then, for each $k$mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, P ($S|M$). This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

It is worth remarking that GLIMMER builds a non-homogenous Markov model; i.e., different models are created for each of the three codon positions. This type of '3-periodic' Markov chain was introduced in GeneMark (5) to account for patterns that depend on the reading frame.

ALGORITHM AND SYSTEM DESIGN

Setting IMM parameters

In this section we describe how GLIMMER computes the values of the λ parameters for the kth-order IMM described in the preceding section. In addition, we explain the solution to the learning problem mentioned in the introduction. First, a set of known coding sequences must be assembled into a training set. To be certain these are truly coding is somewhat problematic for a new genome. The solution we have adopted is to use only very long orfs and sequences with homology to known genes from other organisms. These can easily be identified *a priori* without knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k + 1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \ldots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

enough data. For a 6th-order model, which requires probabilities for all 7mers, there are a substantial number of 7mers that do not occur sufficiently often, and for 7th and 8th-order models the problem is worse. However, even for 8th-order models, there are some oligomers that occur often enough to be extremely useful predictors. We would like a Markov model that uses these higher-order statistics whenever sufficient data is available. This is one of the key advantages of using an IMM. [Note that there exist other techniques to incorporate variable length predictive models (7,8). We experimented with these alternatives before converging on the approach described here.]

To be more precise, an IMM uses a combination of all the probabilities based on 0, 1, 2, ..., k previous bases, where k is a parameter given to the algorithm. In GLIMMER, we use $k = 8$. Thus for oligomers that occur frequently, the IMM can use an 8th-order model, while it might use a 5th or even lower-order model for rare oligomers. In order to 'smooth' its predictions, an IMM uses predictions from the lower-order models, where much more data is available, to adjust the predictions made from higher-order models.

During training, GLIMMER computes the probability of each base a, c, g, t, following all $k$mers for $0 \leq k \leq 8$. Then, for each $k$mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, P ($S|M$). This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

It is worth remarking that GLIMMER builds a non-homogenous Markov model; i.e., different models are created for each of the three codon positions. This type of '3-periodic' Markov chain was introduced in GeneMark (5) to account for patterns that depend on the reading frame.

ALGORITHM AND SYSTEM DESIGN

Setting IMM parameters

In this section we describe how GLIMMER computes the values of the λ parameters for the kth-order IMM described in the preceding section. In addition, we explain the solution to the learning problem mentioned in the introduction. First, a set of known coding sequences must be assembled into a training set. To be certain these are truly coding is somewhat problematic for a new genome. The solution we have adopted is to use only very long orfs and sequences with homology to known genes from other organisms. These can easily be identified *a priori* without knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k+1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \ldots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

enough data. For a 6th-order model, which requires probabilities for all 7mers, there are a substantial number of 7mers that do not occur sufficiently often, and for 7th and 8th-order models the problem is worse. However, even for 8th-order models, there are some oligomers that occur often enough to be extremely useful predictors. We would like a Markov model that uses these higher-order statistics whenever sufficient data is available. This is one of the key advantages of using an IMM. [Note that there exist other techniques to incorporate variable length predictive models (7,8). We experimented with these alternatives before converging on the approach described here.]

To be more precise, an IMM uses a combination of all the probabilities based on 0, 1, 2, ..., k previous bases, where k is a parameter given to the algorithm. In GLIMMER, we use $k = 8$. Thus for oligomers that occur frequently, the IMM can use an 8th-order model, while it might use a 5th or even lower-order model for rare oligomers. In order to 'smooth' its predictions, an IMM uses predictions from the lower-order models, where much more data is available, to adjust the predictions made from higher-order models.

During training, GLIMMER computes the probability of each base a, c, g, t, following all $k$mers for $0 \leq k \leq 8$. Then, for each $k$mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, P ($S|M$). This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

It is worth remarking that GLIMMER builds a non-homogenous Markov model; i.e., different models are created for each of the three codon positions. This type of '3-periodic' Markov chain was introduced in GeneMark (5) to account for patterns that depend on the reading frame.

ALGORITHM AND SYSTEM DESIGN

Setting IMM parameters

In this section we describe how GLIMMER computes the values of the λ parameters for the kth-order IMM described in the preceding section. In addition, we explain the solution to the learning problem mentioned in the introduction. First, a set of known coding sequences must be assembled into a training set. To be certain these are truly coding is somewhat problematic for a new genome. The solution we have adopted is to use only very long orfs and sequences with homology to known genes from other organisms. These can easily be identified *a priori* without knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k + 1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \ldots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

where S_x is the oligomer ending at position x, and n is the length of the sequence. IMM_8 (S_x), the 8th-order interpolated Markov model score, is computed as

$$\mathrm{IMM}_k(S_x) = \lambda_k(S_{x-1}) \bullet P_k(S_x) + [1 - \lambda_k(S_{x-1})] \bullet \mathrm{IMM}_{k-1}(S_x)$$

where $\lambda_k(S_{x-1})$ is the numeric weight associated with the $k$mer ending at position $x-1$ in the sequence S and $P_k(S_x)$ is the estimate obtained from the training data of the probability of the base located at x in the kth-order model. Thus, the 8th-order IMM score of an oligomer is a linear combination of the predictions made by the 8th, 7th and lesser-order models all the way down to the 0th-order model, which is just the simple prior probabilities of a, c, g, t. The above equation is the solution to the evaluation problem mentioned in the introduction.

From this definition, it is clear that an IMM is in principle always preferable to a fixed-order Markov model. For example, by giving zero weights to all oligomers except 5mers, an IMM will perform identically to a 5th-order Markov model. However, if there are any 6mers that occur frequently enough in the training data to be useful, and if these 6mers predict a different distribution of bases than the corresponding 5mers, then the IMM will outperform the 5th-order model. Not only longer but also shorter oligomers will help improve performance: even if a 5th-order model is better than a 4th-order model, there may be some rare 5mers for which insufficient data are available. A 5th-order model has no choice but to use the unreliable predictions from these rare 5mers, but an IMM can fall back on the much more reliable predictions made by the 4mers in such cases. The experiments described below indicate that both of these phenomena occur and both serve to give IMMs an advantage over fixed-order Markov models.

base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

$$P_i(S_x) = P(s_x|S_{x,i}) = \frac{f(S_{x,i})}{\sum_{b\in\{acgt\}} f(S_{x,i}, b)}$$

The value of $\lambda_i(S_x)$ that we associate with $P_i(S_x)$ can be regarded as a measure of our confidence in the accuracy of this value as an estimate of the true probability. GLIMMER uses two criteria to determine $\lambda_i(S_x)$. The first of these is simply frequency of occurrence. If the number of occurrences of context string $S_{x,i}$ in the training data exceeds a specific threshold value, then $\lambda_i(S_x)$ is set to 1.0. Thus, when there are sufficiently many sample occurrences of a context string in the training data, then those sample probabilities are used. The current default value for this threshold in GLIMMER is 400, which gives ~95% confidence that the sample probabilities are within ±0.05 of the true probabilities from which the sample was taken. (Other thresholds were tested experimentally, but none provided any noticeable improvement.)

When there are insufficiently many sample occurrences of a context string to estimate the probability of the next base with confidence, we employ an additional criterion to assign a λ value. For a given context string $S_{x,i}$ of length i, we compare the observed frequencies of the following base, $f(S_{x,i}, \mathrm{a})$, $f(S_{x,i}, \mathrm{c})$, $f(S_{x,i}, \mathrm{g})$ and $f(S_{x,i}, \mathrm{t})$, with the previously calculated IMM probabilities using the next shorter context, IMM_{i-1} ($S_{x,i-1}$, a), IMM_{i-1} ($S_{x,i-1}$, c), IMM_{i-1} ($S_{x,i-1}$, g) and IMM_{i-1} ($S_{x,i-1}$, t). Using a X^2 test, we determine how likely it is that the four observed frequencies are consistent with the IMM values from the next shorter context. When the frequencies differ significantly from the IMM values, we prefer to use them as better predictors of the next base, i.e., give them a higher λ value. Conversely, when the frequencies are consistent with the IMM values, they offer little predictive value and hence we give them a

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

where S_x is the oligomer ending at position x, and n is the length of the sequence. $\mathrm{IMM}_8(S_x)$, the 8th-order interpolated Markov model score, is computed as

$$\mathrm{IMM}_k(S_x) = \lambda_k(S_{x-1}) \bullet P_k(S_x) + [1 - \lambda_k(S_{x-1})] \bullet \mathrm{IMM}_{k-1}(S_x)$$

where $\lambda_k(S_{x-1})$ is the numeric weight associated with the $k$mer ending at position $x-1$ in the sequence S and $P_k(S_x)$ is the estimate obtained from the training data of the probability of the base located at x in the kth-order model. Thus, the 8th-order IMM score of an oligomer is a linear combination of the predictions made by the 8th, 7th and lesser-order models all the way down to the 0th-order model, which is just the simple prior probabilities of a, c, g, t. The above equation is the solution to the evaluation problem mentioned in the introduction.

From this definition, it is clear that an IMM is in principle always preferable to a fixed-order Markov model. For example, by giving zero weights to all oligomers except 5mers, an IMM will perform identically to a 5th-order Markov model. However, if there are any 6mers that occur frequently enough in the training data to be useful, and if these 6mers predict a different distribution of bases than the corresponding 5mers, then the IMM will outperform the 5th-order model. Not only longer but also shorter oligomers will help improve performance: even if a 5th-order model is better than a 4th-order model, there may be some rare 5mers for which insufficient data are available. A 5th-order model has no choice but to use the unreliable predictions from these rare 5mers, but an IMM can fall back on the much more reliable predictions made by the 4mers in such cases. The experiments described below indicate that both of these phenomena occur and both serve to give IMMs an advantage over fixed-order Markov models.

base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

$$P_i(S_x) = P(s_x|S_{x,i}) = \frac{f(S_{x,i})}{\sum_{b\in\{acgt\}} f(S_{x,i}, b)}$$

The value of $\lambda_i(S_x)$ that we associate with $P_i(S_x)$ can be regarded as a measure of our confidence in the accuracy of this value as an estimate of the true probability. GLIMMER uses two criteria to determine $\lambda_i(S_x)$. The first of these is simply frequency of occurrence. If the number of occurrences of context string $S_{x,i}$ in the training data exceeds a specific threshold value, then $\lambda_i(S_x)$ is set to 1.0. Thus, when there are sufficiently many sample occurrences of a context string in the training data, then those sample probabilities are used. The current default value for this threshold in GLIMMER is 400, which gives ~95% confidence that the sample probabilities are within ±0.05 of the true probabilities from which the sample was taken. (Other thresholds were tested experimentally, but none provided any noticeable improvement.)

When there are insufficiently many sample occurrences of a context string to estimate the probability of the next base with confidence, we employ an additional criterion to assign a λ value. For a given context string $S_{x,i}$ of length i, we compare the observed frequencies of the following base, $f(S_{x,i}, \mathrm{a})$, $f(S_{x,i}, \mathrm{c})$, $f(S_{x,i}, \mathrm{g})$ and $f(S_{x,i}, \mathrm{t})$, with the previously calculated IMM probabilities using the next shorter context, $\mathrm{IMM}_{i-1}(S_{x,i-1}, \mathrm{a})$, $\mathrm{IMM}_{i-1}(S_{x,i-1}, \mathrm{c})$, $\mathrm{IMM}_{i-1}(S_{x,i-1}, \mathrm{g})$ and $\mathrm{IMM}_{i-1}(S_{x,i-1}, \mathrm{t})$. Using a X^2 test, we determine how likely it is that the four observed frequencies are consistent with the IMM values from the next shorter context. When the frequencies differ significantly from the IMM values, we prefer to use them as better predictors of the next base, i.e., give them a higher λ value. Conversely, when the frequencies are consistent with the IMM values, they offer little predictive value and hence we give them a

enough data. For a 6th-order model, which requires probabilities for all 7mers, there are a substantial number of 7mers that do not occur sufficiently often, and for 7th and 8th-order models the problem is worse. However, even for 8th-order models, there are some oligomers that occur often enough to be extremely useful predictors. We would like a Markov model that uses these higher-order statistics whenever sufficient data is available. This is one of the key advantages of using an IMM. [Note that there exist other techniques to incorporate variable length predictive models (7,8). We experimented with these alternatives before converging on the approach described here.]

To be more precise, an IMM uses a combination of all the probabilities based on 0, 1, 2, ..., k previous bases, where k is a parameter given to the algorithm. In GLIMMER, we use $k = 8$. Thus for oligomers that occur frequently, the IMM can use an 8th-order model, while it might use a 5th or even lower-order model for rare oligomers. In order to 'smooth' its predictions, an IMM uses predictions from the lower-order models, where much more data is available, to adjust the predictions made from higher-order models.

During training, GLIMMER computes the probability of each base a, c, g, t, following all $k$mers for $0 \leq k \leq 8$. Then, for each $k$mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, P $(S|M)$. This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

It is worth remarking that GLIMMER builds a non-homogenous Markov model; i.e., different models are created for each of the three codon positions. This type of '3-periodic' Markov chain was introduced in GeneMark (5) to account for patterns that depend on the reading frame.

ALGORITHM AND SYSTEM DESIGN

Setting IMM parameters

In this section we describe how GLIMMER computes the values of the λ parameters for the kth-order IMM described in the preceding section. In addition, we explain the solution to the learning problem mentioned in the introduction. First, a set of known coding sequences must be assembled into a training set. To be certain these are truly coding is somewhat problematic for a new genome. The solution we have adopted is to use only very long orfs and sequences with homology to known genes from other organisms. These can easily be identified *a priori* without knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k + 1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \ldots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

enough data. For a 6th-order model, which requires probabilities for all 7mers, there are a substantial number of 7mers that do not occur sufficiently often, and for 7th and 8th-order models the problem is worse. However, even for 8th-order models, there are some oligomers that occur often enough to be extremely useful predictors. We would like a Markov model that uses these higher-order statistics whenever sufficient data is available. This is one of the key advantages of using an IMM. [Note that there exist other techniques to incorporate variable length predictive models (7,8). We experimented with these alternatives before converging on the approach described here.]

To be more precise, an IMM uses a combination of all the probabilities based on 0, 1, 2, ..., k previous bases, where k is a parameter given to the algorithm. In GLIMMER, we use $k = 8$. Thus for oligomers that occur frequently, the IMM can use an 8th-order model, while it might use a 5th or even lower-order model for rare oligomers. In order to 'smooth' its predictions, an IMM uses predictions from the lower-order models, where much more data is available, to adjust the predictions made from higher-order models.

During training, GLIMMER computes the probability of each base a, c, g, t, following all $k$mers for $0 \leq k \leq 8$. Then, for each $k$mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, P $(S|M)$. This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

It is worth remarking that GLIMMER builds a non-homogenous Markov model; i.e., different models are created for each of the three codon positions. This type of '3-periodic' Markov chain was introduced in GeneMark (5) to account for patterns that depend on the reading frame.

ALGORITHM AND SYSTEM DESIGN

Setting IMM parameters

In this section we describe how GLIMMER computes the values of the λ parameters for the kth-order IMM described in the preceding section. In addition, we explain the solution to the learning problem mentioned in the introduction. First, a set of known coding sequences must be assembled into a training set. To be certain these are truly coding is somewhat problematic for a new genome. The solution we have adopted is to use only very long orfs and sequences with homology to known genes from other organisms. These can easily be identified *a priori* without knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k + 1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \dots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \dots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

During training, GLIMMER computes the probability of each base a, c, g, t, following all $k$mers for $0 \leq k \leq 8$. Then, for each $k$mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, $P(S|M)$. This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

where S_x is the oligomer ending at position x, and n is the length of the sequence. $\mathrm{IMM}_8(S_x)$, the 8th-order interpolated Markov model score, is computed as

$$\mathrm{IMM}_k(S_x) = \lambda_k(S_{x-1}) \bullet P_k(S_x) + [1 - \lambda_k(S_{x-1})] \bullet \mathrm{IMM}_{k-1}(S_x)$$

where $\lambda_k(S_{x-1})$ is the numeric weight associated with the $k$mer ending at position $x-1$ in the sequence S and $P_k(S_x)$ is the estimate obtained from the training data of the probability of the base located at x in the kth-order model. Thus, the 8th-order IMM score of an oligomer is a linear combination of the predictions made by the 8th, 7th and lesser-order models all the way down to the 0th-order model, which is just the simple prior probabilities of a, c, g, t. The above equation is the solution to the evaluation problem mentioned in the introduction.

From this definition, it is clear that an IMM is in principle always preferable to a fixed-order Markov model. For example, by giving zero weights to all oligomers except 5mers, an IMM will perform identically to a 5th-order Markov model. However, if there are any 6mers that occur frequently enough in the training data to be useful, and if these 6mers predict a different distribution of bases than the corresponding 5mers, then the IMM will outperform the 5th-order [illegible]

knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k+1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \ldots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

$$P_i(S_x) = P(s_x|S_{x,i}) = \frac{f(S_{x,i})}{\sum_{b \in \{acgt\}} f(S_{x,i}, b)}$$

The value of $\lambda_i(S_x)$ that we associate with $P_i(S_x)$ can be regarded as a measure of our confidence in the accuracy of this value as an estimate of the true probability. GLIMMER uses two criteria to determine $\lambda_i(S_x)$. The first of these is simply frequency of occurrence. If the number of occurrences of context string $S_{x,i}$ in the training data exceeds a specific threshold value, then $\lambda_i(S_x)$ is set to 1.0. Thus, when there are sufficiently many sample occurrences of a context string in the training data, then those sample probabilities are used. The current default value for this threshold in GLIMMER is 400, which gives ~95% confidence that the sample probabilities are within ±0.05 of the true probabilities from which the sample was taken. (Other thresholds were tested experimentally, but none provided any noticeable improvement.)

When there are insufficiently many sample occurrences of a context string to estimate the probability of the next base with confidence, we employ an additional criterion to assign a λ value. For a given context string $S_{x,i}$ of length i, we compare the observed frequencies of the following base, $f(S_{x,i}, \mathrm{a})$, $f(S_{x,i}, \mathrm{c})$, $f(S_{x,i}, \mathrm{g})$ and [illegible]

During training, GLIMMER computes the probability of each base a, c, g, t, following all *k*mers for $0 \leq k \leq 8$. Then, for each *k*mer it computes a weight to use in combining the predictions of different order models. Details of the algorithm for computing these weights are given in the Algorithm and system design section. Once the weights are computed, GLIMMER evaluates new sequences by computing the probability that the model M generated the sequence S, P $(S|M)$. This probability is computed as

$$P(S|M) = \sum_{x=1}^{n} \mathbf{IMM}_8(S_x)$$

where S_x is the oligomer ending at position x, and n is the length of the sequence. IMM_8 (S_x), the 8th-order interpolated Markov model score, is computed as

$$\mathrm{IMM}_k(S_x) = \lambda_k(S_{x-1}) \bullet P_k(S_x) + [1 - \lambda_k(S_{x-1})] \bullet \mathrm{IMM}_{k-1}(S_x)$$

where $\lambda_k(S_{x-1})$ is the numeric weight associated with the *k*mer ending at position $x - 1$ in the sequence S and $P_k(S_x)$ is the estimate obtained from the training data of the probability of the base located at x in the kth-order model. Thus, the 8th-order IMM score of an oligomer is a linear combination of the predictions made by the 8th, 7th and lesser-order models all the way down to the 0th-order model, which is just the simple prior probabilities of a, c, g, t. The above equation is the solution to the evaluation problem mentioned in the introduction.

From this definition, it is clear that an IMM is in principle always preferable to a fixed-order Markov model. For example, by giving zero weights to all oligomers except 5mers, an IMM will perform identically to a 5th-order Markov model. However, if there are any 6mers that occur frequently enough in the training data to be useful, and if these 6mers predict a different distribution of bases than the corresponding 5mers, then the IMM will outperform the 5th-order

knowing anything else about the genome being analyzed.

From the training set of genes, the frequencies of occurrence of all possible substring patterns of length 1 to $k + 1$ are tabulated in each of the six reading frames. (The last base in the substring defines the reading frame.) For simplicity, let us consider just a single reading frame and use $f(S)$ to denote the number of occurrences of string (sequence) $S = s_1 s_2 \ldots s_n$. (This same procedure is repeated for each of the six reading frames.) From these frequencies we get initial estimates of the probability of base s_x occurring given the context string $s_{x-i}, s_{x-i+1}, \ldots, s_{x-1}$, denoted by $S_{x,i}$ (i.e., the i bases just previous to position x). We compute the probability of base s_x given the i previous bases as

$$P_i(S_x) = P(s_x|S_{x,i}) = \frac{f(S_{x,i})}{\sum_{b\in\{acgt\}} f(S_{x,i}, b)}$$

The value of $\lambda_i(S_x)$ that we associate with $P_i(S_x)$ can be regarded as a measure of our confidence in the accuracy of this value as an estimate of the true probability. GLIMMER uses two criteria to determine $\lambda_i(S_x)$. The first of these is simply frequency of occurrence. If the number of occurrences of context string $S_{x,i}$ in the training data exceeds a specific threshold value, then $\lambda_i(S_x)$ is set to 1.0. Thus, when there are sufficiently many sample occurrences of a context string in the training data, then those sample probabilities are used. The current default value for this threshold in GLIMMER is 400, which gives ~95% confidence that the sample probabilities are within ±0.05 of the true probabilities from which the sample was taken. (Other thresholds were tested experimentally, but none provided any noticeable improvement.)

When there are insufficiently many sample occurrences of a context string to estimate the probability of the next base with confidence, we employ an additional criterion to assign a λ value. For a given context string $S_{x,i}$ of length i, we compare the observed frequencies of the following base, $f(S_{x,i}, \mathrm{a})$, $f(S_{x,i}, \mathrm{c})$, $f(S_{x,i}, \mathrm{g})$ and where S_x is the oligomer ending at position x, and n is the length of the sequence. IMM_8 (S_x), the 8th-order interpolated Markov model score, is computed as

$$\text{IMM}_k(S_x) = \lambda_k(S_{x-1}) \bullet P_k(S_x) + [1 - \lambda_k(S_{x-1})] \bullet \text{IMM}_{k-1}(S_x)$$

where $\lambda_k(S_{x-1})$ is the numeric weight associated with the $k$mer ending at position $x - 1$ in the sequence S and $P_k(S_x)$ is the estimate obtained from the training data of the probability of the base located at x in the kth-order model. Thus, the 8th-order IMM score of an oligomer is a linear combination of the predictions made by the 8th, 7th and lesser-order models all the way down to the 0th-order model, which is just the simple prior probabilities of a, c, g, t. The above equation is the solution to the evaluation problem mentioned in the introduction.

From this definition, it is clear that an IMM is in principle always preferable to a fixed-order Markov model. For example, by giving zero weights to all oligomers except 5mers, an IMM will perform identically to a 5th-order Markov model. However, if there are any 6mers that occur frequently enough in the training data to be useful, and if these 6mers predict a different distribution of bases than the corresponding 5mers, then the IMM will outperform the 5th-order model. Not only longer but also shorter oligomers will help improve performance: even if a 5th-order model is better than a 4th-order model, there may be some rare 5mers for which insufficient data are available. A 5th-order model has no choice but to use the unreliable predictions from these rare 5mers, but an IMM can fall back on the much more reliable predictions made by the 4mers in such cases. The experiments described below indicate that both of these phenomena occur and both serve to give IMMs an advantage over fixed-order Markov models.

$$P_i(S_x) = P(s_x | S_{x,i}) = \frac{f(S_{x,i})}{\sum_{b \in \{acgt\}} f(S_{x,i}, b)}$$

The value of $\lambda_i(S_x)$ that we associate with $P_i(S_x)$ can be regarded as a measure of our confidence in the accuracy of this value as an estimate of the true probability. GLIMMER uses two criteria to determine $\lambda_i(S_x)$. The first of these is simply frequency of occurrence. If the number of occurrences of context string $S_{x,i}$ in the training data exceeds a specific threshold value, then $\lambda_i(S_x)$ is set to 1.0. Thus, when there are sufficiently many sample occurrences of a context string in the training data, then those sample probabilities are used. The current default value for this threshold in GLIMMER is 400, which gives ~95% confidence that the sample probabilities are within ±0.05 of the true probabilities from which the sample was taken. (Other thresholds were tested experimentally, but none provided any noticeable improvement.)

When there are insufficiently many sample occurrences of a context string to estimate the probability of the next base with confidence, we employ an additional criterion to assign a λ value. For a given context string $S_{x,i}$ of length i, we compare the observed frequencies of the following base, $f(S_{x,i}, \text{a})$, $f(S_{x,i}, \text{c})$, $f(S_{x,i}, \text{g})$ and $f(S_{x,i}, \text{t})$, with the previously calculated IMM probabilities using the next shorter context, IMM_{i-1} ($S_{x,i-1}$, a), IMM_{i-1} ($S_{x,i-1}$, c), IMM_{i-1} ($S_{x,i-1}$, g) and IMM_{i-1} ($S_{x,i-1}$, t). Using a X^2 test, we determine how likely it is that the four observed frequencies are consistent with the IMM values from the next shorter context. When the frequencies differ significantly from the IMM values, we prefer to use them as better predictors of the next base, i.e., give them a higher λ value. Conversely, when the frequencies are consistent with the IMM values, they offer little predictive value and hence we give them a lower λ value. Specifically, we calculate the χ^2 confidence c that the frequencies are not consistent with the IMM probabilities and set

$$\lambda_i(S_{x-1}) = \begin{cases} 0.0 & \text{if } c < 0.50 \\ \frac{c}{400} \sum_{b \in \{acgt\}} f(s_1 s_2 \ldots s_i b) & \text{if } c \geq 0.50 \end{cases}$$

Thus, we assign higher λ values based on a combination of predictive value, determined by χ^2 significance, and accuracy, determined by frequency of occurrence. This λ value now defines the probabilities IMM_i ($S_{x,i}$, b) for $b \in$ {a, c, g, t} according to equation 1. [Other methods of assigning λ values for IMMs have been developed (9,10). We experimented with these methods in addition to the one described above, and comparative results will be given in a follow up paper. Roberts (11), cited in (12) also describes a method for building nonuniform Markov models.]

The GLIMMER system

The GLIMMER system consists of two programs. The first of these, called build-imm, takes an input set of sequences and builds and outputs the IMM for them as described above. These sequences can be complete genes or just partial orfs. The second program, called glimmer, then uses this IMM to identify putative genes in an entire genome. Glimmer does not use sliding windows to score regions. Instead, it first identifies all orfs longer than some specified threshold value, and scores each one in all six reading frames. Those that score higher than a designated threshold in the correct reading frame are then selected for further processing. These selected orfs are then examined for overlaps. If two orfs in different reading frames overlap (by more than some designated minimum length), the overlapping region alone is scored separately. The overlap region's six reading frame scores are then compared with those of the two overlapping orfs to see which frame scores highest. In general, when a longer orf overlaps a shorter orf and the overlap region scores

both of these criteria: (i) the orf is >500 bases long, which provides the basis for a statistical argument that the gene is highly likely to be a coding region, since orfs of this length almost never occur in non-coding DNA. (ii) The orf does not overlap any other orf longer than 500 bp. Using these criteria, we were able to collect 1168 orfs from the current version of *H.influenzae* (GenBank accession L42023), which contains 1717 annotated genes. Thirty-two of these did not match CDS entries, but we included them anyway. This gives us a completely automatic training procedure for GLIMMER, requiring no human intervention.

This experiment compared GLIMMER's IMM to a conventional fixed-length Markov model on the *H.influenzae* genome data. We followed identical training protocols for both the IMM and a fixed-length 5th-order Markov model. [This 5th-order Markov model is the same model as that used by GeneMark (6). Because we did not have access to the GeneMark source code, we could not retrain that system on our data, so we implemented our own model based on published descriptions of GeneMark.] All post-processing to resolve overlaps was also identical for both methods. Thus the only difference was the model itself: in one case an interpolated Markov model, and in the other case a 5th-order Markov model. Note that we also implemented 4th and 6th-order Markov models, but the 5th-order model performed better than these. The results are shown in Table 1.

Table 1. Comparison of the IMM model used in GLIMMER to a 5th-order Markov model

Model	Genes found	Genes missed	Additional genes
GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra

The GLIMMER system

The GLIMMER system consists of two programs. The first of these, called build-imm, takes an input set of sequences and builds and outputs the IMM for them as described above. These sequences can be complete genes or just partial orfs. The second program, called glimmer, then uses this IMM to identify putative genes in an entire genome. Glimmer does not use sliding windows to score regions. Instead, it first identifies all orfs longer than some specified threshold value, and scores each one in all six reading frames. Those that score higher than a designated threshold in the correct reading frame are then selected for further processing. These selected orfs are then examined for overlaps. If two orfs in different reading frames overlap (by more than some designated minimum length), the overlapping region alone is scored separately. The overlap region's six reading frame scores are then compared with those of the two overlapping orfs to see which frame scores highest. In general, when a longer orf overlaps a shorter orf and the overlap region scores highest in the reading frame of the longer orf, then the shorter orf is eliminated as a gene candidate. The final output of the program is a list of putative gene coordinates in the genome, together with notations for each one that may have had a suspicious overlap with another gene candidate. These 'suspect' gene candidates (usually a very small percentage of the total) can then be examined manually to determine if they are in fact genes. Samples of GLIMMER outputs for the *H.pylori* genome are available on the GLIMMER web site at http://www.cs.jhu.edu/labs/compbio/glimmer.html, which also contains results for *E.coli* and *H.influenzae*. The GLIMMER system, including all source code, is freely available from this site.

retrain that system on our data, so we implemented our own model based on published descriptions of GeneMark.] All post-processing to resolve overlaps was also identical for both methods. Thus the only difference was the model itself: in one case an interpolated Markov model, and in the other case a 5th-order Markov model. Note that we also implemented 4th and 6th-order Markov models, but the 5th-order model performed better than these. The results are shown in Table 1.

Table 1. Comparison of the IMM model used in GLIMMER to a 5th-order Markov model

Model	Genes found	Genes missed	Additional genes
GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

Of the 37 genes missed by GLIMMER's IMM, only one was found by the 5th-order model. In contrast, the IMM found 107 genes that the 5th order model missed. For this run, a pre-set threshold prevented both systems from finding genes shorter than 100 bp, and six of the 37 genes missed by GLIMMER were below this threshold. Of the remaining 31 genes, only one was longer than 500 bp. Finally, note that this was a completely 'self-trained' experiment in which database matches were not used for training; augmenting the training set with these additional genes will almost certainly improve performance further. Of the 209 additional genes called by the system, some can be eliminated from consideration by comparison with functional RNA sequences. The remainder may or may not be

The GLIMMER system

The GLIMMER system consists of two programs. The first of these, called build-imm, takes an input set of sequences and builds and outputs the IMM for them as described above. These sequences can be complete genes or just partial orfs. The second program, called glimmer, then uses this IMM to identify putative genes in an entire genome. Glimmer does not use sliding windows to score regions. Instead, it first identifies all orfs longer than some specified threshold value, and scores each one in all six reading frames. Those that score higher than a designated threshold in the correct reading frame are then selected for further processing. These selected orfs are then examined for overlaps. If two orfs in different reading frames overlap (by more than some designated minimum length), the overlapping region alone is scored separately. The overlap region's six reading frame scores are then compared with those of the two overlapping orfs to see which frame scores highest. In general, when a longer orf overlaps a shorter orf and the overlap region scores highest in the reading frame of the longer orf, then the shorter orf is eliminated as a gene candidate. The final output of the program is a list of putative gene coordinates in the genome, together with notations for each one that may have had a suspicious overlap with another gene candidate. These 'suspect' gene candidates (usually a very small percentage of the total) can then be examined manually to determine if they are in fact genes. Samples of GLIMMER outputs for the *H.pylori* genome are available on the GLIMMER web site at http://www.cs.jhu.edu/labs/compbio/glimmer.html, which also contains results for *E.coli* and *H.influenzae*. The GLIMMER system, including all source code, is freely available from this site.

retrain that system on our data, so we implemented our own model based on published descriptions of GeneMark.] All post-processing to resolve overlaps was also identical for both methods. Thus the only difference was the model itself: in one case an interpolated Markov model, and in the other case a 5th-order Markov model. Note that we also implemented 4th and 6th-order Markov models, but the 5th-order model performed better than these. The results are shown in Table 1.

Table 1. Comparison of the IMM model used in GLIMMER to a 5th-order Markov model

Model	Genes found	Genes missed	Additional genes
GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

Of the 37 genes missed by GLIMMER's IMM, only one was found by the 5th-order model. In contrast, the IMM found 107 genes that the 5th order model missed. For this run, a pre-set threshold prevented both systems from finding genes shorter than 100 bp, and six of the 37 genes missed by GLIMMER were below this threshold. Of the remaining 31 genes, only one was longer than 500 bp. Finally, note that this was a completely 'self-trained' experiment in which database matches were not used for training; augmenting the training set with these additional genes will almost certainly improve performance further. Of the 209 additional genes called by the system, some can be eliminated from consideration by comparison with functional RNA sequences. The remainder may or may not be overlap (by more than some designated minimum length), the overlapping region alone is scored separately. The overlap region's six reading frame scores are then compared with those of the two overlapping orfs to see which frame scores highest. In general, when a longer orf overlaps a shorter orf and the overlap region scores highest in the reading frame of the longer orf, then the shorter orf is eliminated as a gene candidate. The final output of the program is a list of putative gene coordinates in the genome, together with notations for each one that may have had a suspicious overlap with another gene candidate. These 'suspect' gene candidates (usually a very small percentage of the total) can then be examined manually to determine if they are in fact genes. Samples of GLIMMER outputs for the *H.pylori* genome are available on the GLIMMER web site at http://www.cs.jhu.edu/labs/compbio/glimmer.html, which also contains results for *E.coli* and *H.influenzae*. The GLIMMER system, including all source code, is freely available from this site.

METHODS AND RESULTS

To evaluate the effectiveness of our IMM, we compared it to a conventional fixed-order model on data from *H.influenzae* genome. As a second confirming test, we ran it on the recently sequenced *H.pylori* genome and did a careful comparison of the genes found by GLIMMER to those annotated in the public databases and to the genes found by the GeneMark system.

Comparison on *H.influenzae*

Haemophilus influenzae has many putative genes whose existence has not been confirmed biologically. For this experiment, we wanted to train GLIMMER using only genes that had a very high likelihood of being real; therefore, we chose for training a set of orfs that satisfy

GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

Of the 37 genes missed by GLIMMER's IMM, only one was found by the 5th-order model. In contrast, the IMM found 107 genes that the 5th order model missed. For this run, a pre-set threshold prevented both systems from finding genes shorter than 100 bp, and six of the 37 genes missed by GLIMMER were below this threshold. Of the remaining 31 genes, only one was longer than 500 bp. Finally, note that this was a completely 'self-trained' experiment in which database matches were not used for training; augmenting the training set with these additional genes will almost certainly improve performance further. Of the 209 additional genes called by the system, some can be eliminated from consideration by comparison with functional RNA sequences. The remainder may or may not be expressed genes, and further biological evidence is required to resolve these genes.

Gene finding accuracy on *H.pylori*

Finally, in a test designed to run the system as it will be used on new, complete genomes, we ran GLIMMER on the complete, recently sequenced genome of *H.pylori* (13), the bacterium that causes stomach ulcers. A training set of brute force orfs that were >500 nt were collected from the complete genome of *H.pylori*. (This training set was collected from the genome without reference to any annotation, exactly as it would be for a brand new sequence.) The resulting IMM model was then compared to the annotated set of genes identified for this organism. The 1590 genes annotated for *Helicobacter* were identified by integrating the following sets of lower λ value. Specifically, we calculate the χ^2 confidence c that the frequencies are not consistent with the IMM probabilities and set

$$\lambda_i(S_{x-1}) = \begin{cases} 0.0 & \text{if } c < 0.50 \\ \frac{c}{400} \sum_{b \in \{acgt\}} f(s_1 s_2 \ldots s_i b) & \text{if } c \geq 0.50 \end{cases}$$

Thus, we assign higher λ values based on a combination of predictive value, determined by χ^2 significance, and accuracy, determined by frequency of occurrence. This λ value now defines the probabilities IMM_i $(S_{x,i}, b)$ for $b \in \{a, c, g, t\}$ according to equation 1. [Other methods of assigning λ values for IMMs have been developed (9,10). We experimented with these methods in addition to the one described above, and comparative results will be given in a follow up paper. Roberts (11), cited in (12) also describes a method for building nonuniform Markov models.]

The GLIMMER system

The GLIMMER system consists of two programs. The first of these, called build-imm, takes an input set of sequences and builds and outputs the IMM for them as described above. These sequences can be complete genes or just partial orfs. The second program, called glimmer, then uses this IMM to identify putative genes in an entire genome. Glimmer does not use sliding windows to score regions. Instead, it first identifies all orfs longer than some specified threshold value, and scores each one in all six reading frames. Those that score higher than a designated threshold in the correct reading frame are

both of these criteria: (i) the orf is >500 bases long, which provides the basis for a statistical argument that the gene is highly likely to be a coding region, since orfs of this length almost never occur in non-coding DNA. (ii) The orf does not overlap any other orf longer than 500 bp. Using these criteria, we were able to collect 1168 orfs from the current version of *H.influenzae* (GenBank accession L42023), which contains 1717 annotated genes. Thirty-two of these did not match CDS entries, but we included them anyway. This gives us a completely automatic training procedure for GLIMMER, requiring no human intervention.

This experiment compared GLIMMER's IMM to a conventional fixed-length Markov model on the *H.influenzae* genome data. We followed identical training protocols for both the IMM and a fixed-length 5th-order Markov model. [This 5th-order Markov model is the same model as that used by GeneMark (6). Because we did not have access to the GeneMark source code, we could not retrain that system on our data, so we implemented our own model based on published descriptions of GeneMark.] All post-processing to resolve overlaps was also identical for both methods. Thus the only difference was the model itself: in one case an interpolated Markov model, and in the other case a 5th-order Markov model. Note that we also implemented 4th and 6th-order Markov models, but the 5th-order model performed better than these. The results are shown in Table 1.

Table 1. Comparison of the IMM model used in GLIMMER to a 5th-order Markov model

value, and scores each one in all six reading frames. Those that score higher than a designated threshold in the correct reading frame are then selected for further processing. These selected orfs are then examined for overlaps. If two orfs in different reading frames overlap (by more than some designated minimum length), the overlapping region alone is scored separately. The overlap region's six reading frame scores are then compared with those of the two overlapping orfs to see which frame scores highest. In general, when a longer orf overlaps a shorter orf and the overlap region scores highest in the reading frame of the longer orf, then the shorter orf is eliminated as a gene candidate. The final output of the program is a list of putative gene coordinates in the genome, together with notations for each one that may have had a suspicious overlap with another gene candidate. These 'suspect' gene candidates (usually a very small percentage of the total) can then be examined manually to determine if they are in fact genes. Samples of GLIMMER outputs for the *H.pylori* genome are available on the GLIMMER web site at http://www.cs.jhu.edu/labs/compbio/glimmer.html, which also contains results for *E.coli* and *H.influenzae*. The GLIMMER system, including all source code, is freely available from this site.

METHODS AND RESULTS

To evaluate the effectiveness of our IMM, we compared it to a conventional fixed-order model on data from *H.influenzae* genome. As a second confirming test, we ran it on the recently sequenced *H.pylori* genome and did a careful comparison of the genes found by GLIMMER to those annotated in the public databases and to the genes found by the GeneMark system.

Comparison on *H.influenzae*

Table 1. Comparison of the IMM model used in GLIMMER to a 5th-order Markov model

Model	Genes found	Genes missed	Additional genes
GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

Of the 37 genes missed by GLIMMER's IMM, only one was found by the 5th-order model. In contrast, the IMM found 107 genes that the 5th order model missed. For this run, a pre-set threshold prevented both systems from finding genes shorter than 100 bp, and six of the 37 genes missed by GLIMMER were below this threshold. Of the remaining 31 genes, only one was longer than 500 bp. Finally, note that this was a completely 'self-trained' experiment in which database matches were not used for training; augmenting the training set with these additional genes will almost certainly improve performance further. Of the 209 additional genes called by the system, some can be eliminated from consideration by comparison with functional RNA sequences. The remainder may or may not be expressed genes, and further biological evidence is required to resolve these genes.

Gene finding accuracy on *H.pylori*

Finally, in a test designed to run the system as it will be used on new, complete genomes, we ran GLIMMER on the complete, recently sequenced genome of *H.pylori* (13), the bacterium that causes stomach ulcers. A training set of brute force orfs that were >500 nt were collected from the complete genome of *H.pylori*. (This training set was collected from the genome without reference to any overlap (by more than some designated minimum length), the overlapping region alone is scored separately. The overlap region's six reading frame scores are then compared with those of the two overlapping orfs to see which frame scores highest. In general, when a longer orf overlaps a shorter orf and the overlap region scores highest in the reading frame of the longer orf, then the shorter orf is eliminated as a gene candidate. The final output of the program is a list of putative gene coordinates in the genome, together with notations for each one that may have had a suspicious overlap with another gene candidate. These 'suspect' gene candidates (usually a very small percentage of the total) can then be examined manually to determine if they are in fact genes. Samples of GLIMMER outputs for the *H.pylori* genome are available on the GLIMMER web site at http://www.cs.jhu.edu/labs/compbio/glimmer.html[!], which also contains results for *E.coli* and *H.influenzae*. The GLIMMER system, including all source code, is freely available from this site.

METHODS AND RESULTS

To evaluate the effectiveness of our IMM, we compared it to a conventional fixed-order model on data from *H.influenzae* genome. As a second confirming test, we ran it on the recently sequenced *H.pylori* genome and did a careful comparison of the genes found by GLIMMER to those annotated in the public databases and to the genes found by the GeneMark system.

Comparison on *H.influenzae*

Haemophilus influenzae has many putative genes whose existence has not been confirmed biologically. For this experiment, we wanted to train GLIMMER using only genes that had a very high likelihood of being real; therefore, we chose for training a set of orfs that satisfy

GLIMMER IMM	1680 (97.8%	37	209
5th-Order Markov	1574 (91.7%)	143	104

The first column indicates how many of the 1717 annotated genes in *H.influenzae* were found by each algorithm. The 'additional genes' column shows how many extra genes, not included in the 1717 annotated entries, were called genes by each method.

Of the 37 genes missed by GLIMMER's IMM, only one was found by the 5th-order model. In contrast, the IMM found 107 genes that the 5th order model missed. For this run, a pre-set threshold prevented both systems from finding genes shorter than 100 bp, and six of the 37 genes missed by GLIMMER were below this threshold. Of the remaining 31 genes, only one was longer than 500 bp. Finally, note that this was a completely 'self-trained' experiment in which database matches were not used for training; augmenting the training set with these additional genes will almost certainly improve performance further. Of the 209 additional genes called by the system, some can be eliminated from consideration by comparison with functional RNA sequences. The remainder may or may not be expressed genes, and further biological evidence is required to resolve these genes.

Gene finding accuracy on *H.pylori*

Finally, in a test designed to run the system as it will be used on new, complete genomes, we ran GLIMMER on the complete, recently sequenced genome of *H.pylori* (13), the bacterium that causes stomach ulcers. A training set of brute force orfs that were >500 nt were collected from the complete genome of *H.pylori*. (This training set was collected from the genome without reference to any annotation, exactly as it would be for a brand new sequence.) The resulting IMM model was then compared to the annotated set of genes identified for this organism. The 1590 genes annotated for *Helicobacter* were identified by integrating the following sets of information: (i) evaluating brute force orfs for protein-level sequence similarity matches to the public archives, (ii) predicting coding regions using the GeneMark system and (iii) collecting 'intergenic' orfs that were found between the genes with database matches and the genes called by GeneMark. We consider the *H.pylori* sequence annotation to have been intensively evaluated by the research community, and as yet, no unidentified genes have been reported since the *H.pylori* publication.

The annotated genes were compared to the results of the GLIMMER algorithm, and 1548 of the 1590 genes were found to have been correctly identified. An additional 314 potential orfs were found by the system in the *H.pylori* genome. Some of these additional genes can be eliminated by discarding those that conflict with ribosomal and transfer RNAs, but the remainder cannot be ruled out as authentic genes without further biological evidence. The set of 42 unidentified genes, representing a potential false negative rate of 2.6%, were examined further. Nineteen of these genes from the *H.pylori* annotation were under 100 nt in length, and possibly below the length for meaningful detection by compositional methods. Orfs that have matches to proteins in the current public archives serve as the most reliable and independent verification that an orf is an authentic gene; of these orfs, only seven were present in the 42 genes that GLIMMER did not identify. This suggests a minimal false negative rate of 0.44% for GLIMMER.

Note that for this experiment, GLIMMER used a minimum gene length of 90 bp; this length can be changed with a simple command line parameter. With a minimum gene length of 180 bp (60 amino acids), for example, GLIMMER calls 286 fewer genes in *H.pylori*.

Finally, we conducted a limited comparison to the GeneMark system (6). To keep the comparison simple, we only considered the 974 genes from *H.pylori* that had database matches to other organisms; these can safely be considered to be 'true' genes. GLIMMER, was trained exclusively on orfs longer than 500 bp, be re-run repeatedly until it converges. This iterative algorithm will also be available as an option in the GLIMMER system.

CONCLUSION

Evaluating the accuracy of a microbial gene finder is difficult, because the genes annotated in GenBank do not always have biological evidence to back up their existence. As the annotation becomes more stable, more accurate estimates of accuracy will be possible. At the same time, better gene finders should result because the available training data will improve. Although GLIMMER'S sensitivity is nearing 100% already, there are several important areas of future improvements. One is to improve its specificity by reducing the number of false positives (after first confirming that the unannotated genes found by the system are in fact false). Specificity can already be reduced substantially, at the cost of slightly reducing sensitivity, by increasing the minimum length orf that GLIMMER will consider as a gene. Another is to incorporate separate pattern analysis algorithms that will allow the system to find promoters, enhancers, terminators and other signals that occur in intergenic regions. Accurate location of these signals is an important problem in its own right, and a system that integrates the content scoring approach of GLIMMER with a good signal identification algorithm should produce better results than either approach could independently.

ACKNOWLEDGEMENTS

Thanks to Mark Borodovsky and Alexander Lukashin for kindly sharing the results of GeneMarkHMM on the *H.pylori* genome. S.L.S. is supported by the National Human Genome Research Institute at NIH under Grant No. K01-HG00022-1. S.L.S. and A.L.D. are supported by the National Science foundation under Grant No. IRI-9530462. S.K. is supported by NSF IRI-9529227.

Glimmer

README & OUTPUT …