Report Number: WUCSE-2005-12

2005-04-15

An Iterative Learning Algorithm for Deciphering Stegoscripts: a Grammatical Approach for Motif Discovery

Guandong Wang and Weixiong Zhang

Steganography, or information hiding, is to conceal the existence of messages so as to protect their confidentiality. We consider de-ciphering a stegoscript, a text with secret messages embedded within a covertext, and identifying the vocabularies used in the mes-sages, with no knowledge of the vocabularies and grammar in which the script was writ-ten. Our research was motivated by the prob-lem of identifying conserved non-coding func-tional elements (motifs) in regulatory regions of genome sequences, which we view as stego-scripts constructed by nature with a statis-tical model consisting of a dictionary and a grammar. We develop an iterative learning algorithm, WordSpy,... **Read complete abstract on page 2.**

An Iterative Learning Algorithm for Deciphering Stegoscripts: a Grammatical Approach for Motif Discovery

Guandong Wang and Weixiong Zhang

Complete Abstract:

Steganography, or information hiding, is to conceal the existence of messages so as to protect their confidentiality. We consider de-ciphering a stegoscript, a text with secret messages embedded within a covertext, and identifying the vocabularies used in the mes-sages, with no knowledge of the vocabularies and grammar in which the script was writ-ten. Our research was motivated by the prob-lem of identifying conserved non-coding func-tional elements (motifs) in regulatory regions of genome sequences, which we view as stego-scripts constructed by nature with a statis-tical model consisting of a dictionary and a grammar. We develop an iterative learning algorithm, WordSpy, to learn such a model from a stegoscript. The model then can be applied to identify the embedded secret mes-sages, i.e., the functional motifs. Our algo-rithm can successfully recover the most pos-sible text of the first ten chapters of a novel embedded in a stegoscript and identify the transcription factor binding motifs in the up-stream regions of $\sim$ 800 yeast genes.

2005-12

An Iterative Learning Algorithm for Deciphering Stegoscripts: a Grammatical Approach for Motif Discovery

Authors: Wang, Guandong; Zhang, Weixiong

April 15, 2005

Abstract: Steganography, or information hiding, is to conceal the existence of messages so as to protect their confidentiality. We consider deciphering a stegoscript, a text with secret messages embedded within a covertext, and identifying the vocabularies used in the messages, with no knowledge of the vocabularies and grammar in which the script was written. Our research was motivated by the problem of identifying conserved non-coding functional elements (motifs) in regulatory regions of genome sequences, which we view as stegoscripts constructed by nature with a statistical model consisting of a dictionary and a grammar. We develop an iterative learning algorithm, WordSpy, to learn such a model from a stegoscript. The model then can be applied to identify the embedded secret messages, i.e., the functional motifs. Our algorithm can successfully recover the most possible text of the first ten chapters of a novel embedded in a stegoscript and identify the transcription factor binding motifs in the upstream regions of ~800 yeast genes.

An Iterative Learning Algorithm for Deciphering Stegoscripts: a Grammatical Approach for Motif Discovery

Guandong Wang GW2@CSE.WUSTL.EDU
Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA

Weixiong Zhang ZHANG@CSE.WUSTL.EDU
Department of Computer Science and Engineering and Department of Genetics, Washington University in St. Louis, St. Louis, MO 63130, USA

Abstract

Steganography, or information hiding, is to conceal the existence of messages so as to protect their confidentiality. We consider deciphering a stegoscript, a text with secret messages embedded within a covertext, and identifying the vocabularies used in the messages, with no knowledge of the vocabularies and grammar in which the script was written. Our research was motivated by the problem of identifying conserved non-coding functional elements (motifs) in regulatory regions of genome sequences, which we view as stegoscripts constructed by nature with a statistical model consisting of a dictionary and a grammar. We develop an iterative learning algorithm, *WordSpy*, to learn such a model from a stegoscript. The model then can be applied to identify the embedded secret messages, i.e., the functional motifs. Our algorithm can successfully recover the most possible text of the first ten chapters of a novel embedded in a stegoscript and identify the transcription factor binding motifs in the upstream regions of ~ 800 yeast genes.

1. Introduction

Unlike cryptography, which is about concealing the content of messages, steganography is about concealing the existence of messages (Wayner, 2002). Many legends told ancient stories of how steganography was used in historic events; the oldest of this sort is perhaps the one about how the Greeks received warning of Xerxes' intentions from a message underneath the wax of a writing tablet, describing a trick of dotting successive letters in a covertext with secret ink. Steganography also exists in today's reality. USA TODAY reported that Bin Laden and his associates have been "hiding maps and photographs of terrorist targets and posting instructions for terrorist activities on sports chat rooms, bulletin boards and other Web sites" (USA TODAY, 2001).

We are interested in discovering secret messages hidden in stegoscripts. For example, we are interested in recovering a news article embedded in a covertext with no knowledge of the language that the article was written in, i.e., no dictionary nor grammar.

Despite a large body of research on steganography on images (Wayner, 2002), steganography on text is rather thin. One related work considered deciphering a sound-to-character association for a given script and using rough knowledge of the language the script was written (Knight & Yamada, 1999). Note also that there is no covertext to be winnowed out in this problem. The chaffing-and-winnowing communication protocol (Rivest, 1998) is also loosely related. In this protocol, genuine packages were sent interspersed among spurious ones to hide the true messages, making the overall transmission a stego sequence of packages while individual genuine packages are intact.

This research is motivated by the problem of identifying functional motifs in a genome (Durbin et al., 1998). Much work has been done on finding non-coding functional elements, mainly transcriptional factor binding motifs (TFBMs). Most motif-finding approaches, including Gibbs Sampler (Lawrence et al., 1993),

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

MEME (Bailey & Elkan, 1995), Consensus (Hertz & Stormo, 1999) and AlignACE (Hughes et al., 2000), apply local search strategies to find local multiple alignments to fit some statistical models which characterize (unknown) motifs. Despite their success, the statistical models used in these methods are for single motif or a fixed number of motifs. Moreover, the algorithms may get easily trapped into local minima, and are in general too slow to handle large sequences.

Another approach is word-counting based (van Helden et al., 1998; Sinha & Tompa, 2000), in which statistically over-represented words are identified through enumeration. These methods are typically fast and able to handle large sequences, and can identify a large number of putative motifs. However, they usually lack of accurate statistical models and suffer from the problems of producing too many spurious motifs.

A dictionary based approach, recently pioneered by Bussemaker et al. (Bussemaker et al., 2002), introduced the concepts of dictionary and word usage frequencies for constructing sequences. The over-representation of a long word is computed as the weighted average of the short words in the current dictionary which can form partitions of the long word. Although this method is in essence word-counting based, it can filter out many spurious motifs which are over-represented only due to overlapping with some real motifs, resulting in a higher accuracy. However, testing the over-representation of longer words by concatenating the shorter words is problematic, and may miss substantial over-represented motifs.

We approach the problem from a perspective of steganography. We view regulatory genome sequences as a stegoscript in which functional TFBMs are secret messages embedded in and protected by a covertext of background sequences. In other words, we hypothesize that nature has a dictionary of sequence motifs and a complex grammar for constructing a genome. This hypothesis is partially supported by the current understanding that most genomes carry a large amount of "garbage" sequences with no known function. It is very possible that nature uses simple steganography by adding random and redundant sequences as a defence mechanism against possible invasion of foreign viral agents.

In general, we consider the following problem. Given a stego script, discover the vocabularies and grammar with which the script was generated, and recover the original messages in the script. We develop an innovative algorithm, named as WordSpy, that iteratively learns a statistical model with a dictionary and a grammar in the form of a hidden Markov model (HMM). The dictionary consists of conserved words and the grammar (HMM) specifies how the words were used and the steganoscript was created. The statistical model can be finally used to decipher the script.

The idea of modeling DNA sequences by grammars has been around for quite some time (Searls, 1992). Indeed, a grammar-based approach is one of the best for RNA secondary structure prediction (Durbin et al., 1998), and HMMs have been extensively used in gene finding. However, few attempt has been made to motif finding. It is worth to mention that there have been several proposals of using HMM to model motifs (Bailey & Noble, 2003; Sinha et al., 2003; Frith et al., 2001; Xing & Wu, 2003). However, in all of these approaches, motifs are supposed to be given so that the models are predetermined, which in many cases are used to detect motif modules. In our approach, we learn an HMM to fit the script without knowing how the model looks like beforehand, which is obviously a much harder problem.

We apply our algorithm to recovering the first ten chapters of novel *Moby Dick* which were intentionally hidden in a long random string. The results achieved high identification accuracy of more than 80%. We also apply our method to finding TFBMs in the upstream regions of about 800 cell-cycle related genes of *S. cerevisiae* (Spellman et al., 1998). Our algorithm identifies all known yeast cell-cycle TFBMs with high significance, while MobyDick missed quite many of them.

2. Stegoscripts and Statistical Model

A stegoscript is to conceal the secret messages with some covertext. In many cases, the script is generated without any explicit encryption keys. The only knowledge about the script is that secret messages and covertext should have different information contents or statistics, e.g., different word distributions. In some sense, the grammatical model used to generate the script is the key to decipher the script. When the grammatical model is known, deciphering a stegoscript is just as easy as decoding a ciphertext with a given key. Thus to decipher a stegoscript is equivalent to recover the grammatical model.

Usually, the words in the secret messages are more conserved than those in the covertext. This is especially true for genomic regulatory sequences, in which a small number of transcription factors (TFs) mediate a large number of genes (Brivanlou & Darnell, 2002; Lemon & Tjian, 2000), making TFBMs over-represented. Most TFBMs are also conversed across

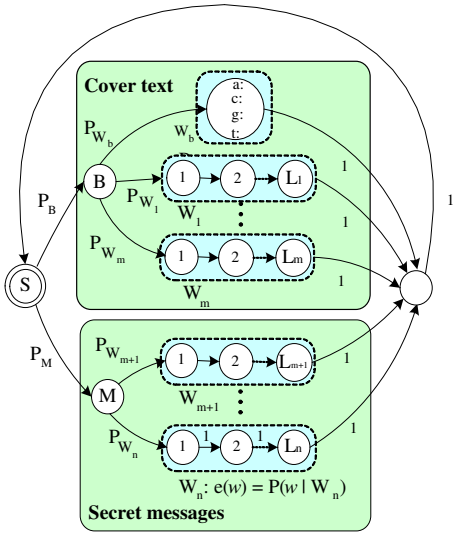


Figure 1. A statistical model (HMM) for generating and deciphering stegoscripts. The model contains two submodels, the *secret message model* generates motifs and the *cover text model* produces covertext with background words. States S, B and M do not emit any letter. A dash box represents a word node, which is actually a combination of several position nodes, each referring to one position in that word. W_b, a only node with single base, always belongs to background model.

closely related species (see, e.g., (Cliften et al., 2003)). Despise their conservations, TFBMs are usually hard to be identified, mainly because there are many repeats (or fake motifs) also over-represented in the regulatory regions. We thus propose a statistical model consisting of a grammar and a dictionary of two sets of words. One type of words in the dictionary are the TFBMs to be identified, and the others are background words in the covertext. The grammar specifies how the words are used to form a stegoscript.

Fig. 1 illustrates our model. It can be used to generate a stegoscript and to decipher it as well. In each step of generating a script, the grammar will choose to produce either a motif word M with probability P_M, or a background word B with probability P_B. Once M is chosen, a (degenerate) motif W_i is randomly drawn from the *motif subdictionary*, with probability P_{W_i}, and an exact word w is generated with probability $P(w|W_i)$. Similar process goes for the background word B too. The generated word is appended to the end of the script and the process repeats until the whole script is created.

The key to deciphering a stegoscript or a set of regulatory sequences is to learn the statistical model with which the script was created. Assume that a stegoscript $\mathcal{S}$ were generated from an unknown model $\langle D^*, G^* \rangle$ of a dictionary D^* and a grammar G^*. Theoretically, we can learn such a model $\langle D, G \rangle = \arg\max_{\langle D', G' \rangle} P(\langle D', G' \rangle | \mathcal{S})$, that most likely generates the script. With no prior knowledge of the true model, the maximum likelihood estimation, $\langle D, G \rangle = \arg\max_{\langle D', G' \rangle} P(\mathcal{S} | \langle D', G' \rangle)$, is a good approximation of $\langle D^*, G^* \rangle$.

In this deciphering problem, we only want to discover the secret messages and do not care about its meanings or how the words in the messages are related. We thus can safely assume that the words are used independently within the grammar. Therefore, a stochastic regular grammar (or HMM) will have enough modeling power to serve the needs of discriminating secret messages from cover text. Obviously we can resort to a higher order grammar if a more accurate model is needed to recover more information, but the computational cost will increase dramatically as well.

3. The Learning Algorithm

The central problem of deciphering a stegoscript is learning a statistical model (an HMM shown in Fig. 1) from the given script. However, this is an extremely difficult problem, as a large number of word nodes are typically unknown and many unknown model parameters to be estimated. It is infeasible and often problematic to directly learn such a model due to the huge search space and abundant of local minima. Therefore, we separate the learning process into two phases, word sampling and model optimization, and adopt an iterative learning strategy similar to EM algorithm (Dempster et al., 1977) to progressively capture short to long words and gradually build such a model. Briefly, letting $\langle D_k, G_k \rangle$ be a statistical model that contains words of lengths up to k (inclusive), in the M step, $\langle D_k, G_k \rangle$ is optimized to best fit the script $\mathcal{S}$; in the E step, the optimized model is used to identify over-represented words of length $k+1$, resulting in a new model $\langle D_{k+1}, G_{k+1} \rangle$.

The overall algorithm starts with the simplest model $\langle D_1, G_1 \rangle$ with only one word of single base in D_1, at the k-th iteration, the algorithm first identifies over-represented words of length k. In this process, the given script $\mathcal{S}$ is assumed to be generated by the current best model $\langle D_{k-1}, G_{k-1} \rangle$. A word is over-represented if it occurs in $\mathcal{S}$ more often then it could be generated by the current model $\langle D_{k-1}, G_{k-1} \rangle$. Furthermore, newly discovered words will be classified as motif words or background words, and merged with D_{k-1} to form the next dictionary D_k. The model is retrofited to accommodate the new words, leading to

the next grammar, G_k. The new model $\langle D_k, G_k\rangle$ is then optimized to fix the script. The overall process repeats until the model covers words up to a maximum length.

The classification of motif words and background words are important to the accuracy of the algorithm. However, when no extra information is available, we can only resort to the over-representation level to select putative motif words. We use Z-score to quantitatively measure the over-representation of a word. The detailed definition and calculation are presented in the following section. With assumption that the covertext usually consists of random sequences, we typically set 3 as the minimum Z-score threshold for a word to be a motif. A higher Z-score means a better chance for a word to be a real motif. As more information is available, more accurate classification can be made.

In the following sections, each phase of the algorithm will be presented in detail.

3.1. Model Optimization

The model optimization is for computing the next model $\langle D_k, G_k\rangle$ based on $\langle D_{k-1}, G_{k-1}\rangle$ to incorporate a set of new over-represented words of length k. The next dictionary D_k can be simply formed by combining D_{k-1} and the new words; the main issue is then to compute the next grammar G_k. G_k has two types of parameters. The first are the transition probabilities, Ψ, corresponding to the word usage frequencies, determining how conserved motifs or background words are used in the given sequences. The second parameters are the emission probabilities, Θ, corresponding to the letter (base) usage frequencies at each position of an over-represented word. As these parameters are unknown, we apply an EM approach to estimate them.

We can write $G_k = (\Psi, \Theta, \mathcal{I})$, where $\Psi = \{P_B, P_M, P_{W_b}, P_{W_1}, P_{W_2}, \cdots, P_{W_n}\}$ is the set of transition probabilities, $\Theta = \{\Theta_b, \Theta_1, \Theta_2, \cdots, \Theta_n\}$ is a set of emission probabilities corresponding to the motifs and words in $D_k = \{W_b, W_1, W_2, \cdots, W_n\}$, and $\mathcal{I} = \{I_{W_i} | W_i \in D_k\}$ is a set of indicators, where

$$I_{W_i} = \begin{cases} 1, \text{if } W_i \text{ is a conserved motif,} \\ 0, \text{if } W_i \text{ is a background word.} \end{cases}$$

I_{W_b} is always set to 0; the other values of $\mathcal{I}$ are determined in the word sampling phase (Section 3.2). That is, $\mathcal{I}$ does not change during model optimization. Without loss of generality, we view a set of sequences as a long sequence $\mathcal{S} = s_1 s_2 \cdots s_q$. Let $\Psi^{(t)}$ and $\Theta^{(t)}$ be G_k's parameters in the t-th iteration of the EM algorithm. The process of model optimization iteratively updates $\Psi^{(t)}$ and $\Theta^{(t)}$ until convergence.

To update $\Psi^{(t)}$ and $\Theta^{(t)}$, we first consider different parses of $\mathcal{S}$. The probability of a parse of $\mathcal{S}$, denoted by ϕ, given $\Psi^{(t)}$ and $\Theta^{(t)}$, can be computed by

$$P(\phi|\mathcal{S}, \Psi^{(t)}, \Theta^{(t)}) = \prod_{W_i \in D} \left(P_{W_i}^{(t)}\right)^{N_{W_i}^{\phi}} \prod_{j=1}^{N_{W_i}^{\phi}} P\left(\chi_{W_i}^{j} | \Theta_i^{(t)}\right)$$

where $N_{W_i}^{\phi}$ is the count (i.e., number of occurrences) of motif W_i in the parse ϕ, and $\chi_{W_i}^{j}$ is the j-th occurrence (or site) of the W_i in $\mathcal{S}$ under ϕ. Then the average number of occurrence of W_i, denoted as N_{W_i}, can be calculated as

$$N_{W_i} = \sum_{\phi \in \Phi} P\left(\phi|\mathcal{S}, \Psi^{(t)}, \Theta^{(t)}\right) N_{W_i}^{\phi} \quad (1)$$

where Φ is the set of all possible parses. The average count of a letter ς at j-th position of W_i, denoted by $C_{W_i}(\varsigma, j)$, can be calculated by

$$C_{W_i}(\varsigma, j) = \sum_{\phi \in \Phi} P\left(\phi|\mathcal{S}, \Psi^{(t)}, \Theta^{(t)}\right) C_{W_i}^{\phi}(\varsigma, j) \quad (2)$$

where $C_{W_i}^{\phi}(\varsigma, j)$ is the count of letter ς at j-th position of W_i in the parse ϕ. Based on maximum likelihood principle, we update the parameters as follows.

$$\begin{cases} P_B^{(t+1)} = \frac{\sum_{W \in D_k} N_W \cdot (1 - I_W)}{\sum_{W \in D_k} N_W}, \\ P_M^{(t+1)} = \frac{\sum_{W \in D_k} N_W \cdot I_W}{\sum_{W \in D_k} N_W}, \\ P_{W_i}^{(t+1)} = \frac{N_{W_i}}{\sum_{W \in D_k} N_W \cdot \delta(I_{W_i}, I_W)}, \\ \Theta_i^{(t+1)}(\varsigma, j) = \frac{C_{W_i}(\varsigma, j)}{\sum_{\varsigma' \in \Sigma} C_{W_i}(\varsigma', j)}, \end{cases} \quad (3)$$

where Σ is the alphabet, $\varsigma \in \Sigma$, $j = 1, \cdots, l(W)$, $l(W)$ is the length of W, and $\delta(x, y)$ equals 1 if $x = y$, or 0 otherwise.

The calculation of (1) and (2) could be costly if we enumerate all possible parses. We adopted the dynamic programming *forward-backward* algorithm (Durbin et al., 1998) to compute the most probable state when observing $s_l \in \mathcal{S}$. More precisely, let π_l be the state of s_l, the probability of s_l being at the j-th position of a motif W under the current grammar can be computed as

$$P(\pi_l = W[j] | \mathcal{S}, G_k) = \frac{f(\mu) \cdot \rho_W \cdot \varrho_W(\mu + 1, \nu) \cdot b(\nu + 1)}{P(\mathcal{S}|G_k)},$$

where W is a degenerate word in D_k, $W[j]$ is the j-th position of W, $f(\mu)$ is the probability of observing $\mathcal{S}$ up to s_μ (inclusive) given G_k, $\mu = l - k$, $\rho_W = P_W(I_W P_M + (1 - I_W)P_B)$, $\varrho_W(i, j) = P(\mathcal{S}_{[i,j]}|W)$,

$b(\nu+1)$ is the probability of observing $\mathcal{S}$ from $s_{\nu+1}$ (inclusive) down to the end of $\mathcal{S}$, and $\nu = l-k+l(W)$. Function $f(i)$ can be recursively computed as

$$f(i) = \sum_{W \in D} \rho_W \cdot \varrho_W(i - l(W) + 1, i) \cdot f(i - l(W)).$$

Similarly $b(i)$ can be computed as

$$b(i) = \sum_{W \in D} \rho_W \cdot \varrho_W(i, i + l(W) - 1) \cdot b(i + l(W)).$$

Evidently, $P(\mathcal{S}|G_k) = f(q) = b(1)$.

Suppose $P(\pi_l = W_i[j]|\mathcal{S}, \Psi^{(t)}, \Theta^{(t)})$ is the probability of observing s_l at the j-th position of a motif W_i given $\Psi^{(t)}$ and $\Theta^{(t)}$, equations (1) and (2) can be simply computed as

$$\begin{cases} N_{W_i} = \sum_{l=1}^{q} P\left(\pi_l = W_i[1]|\mathcal{S}, \Psi^{(t)}, \Theta^{(t)}\right), \\ C_{W_i}(\varsigma, j) = \sum_{l=1}^{q} P\left(\pi_l = W_i[j], s_l = \varsigma|\mathcal{S}, \Psi^{(t)}, \Theta^{(t)}\right), \end{cases}$$

where q is the length of $\mathcal{S}$.

The model optimization is done iteratively using equations in (3) until convergence. This procedure is the most time consuming part of the WordSpy algorithm. Nonetheless, the hash scheme of indexing a word w directly to the degenerate words that may emit w in the dictionary reduces the average computation of the forward-backward algorithm from $O(LN)$ to $O(L)$, with a penalty of space increment of $O(N)$, where L is the sequence length and N the size of the dictionary. The overall space complexity is $O(L+N)$.

3.2. Word Sampling

In the word sampling phase, the algorithm identifies over-represented words of length k based on the optimal model $\langle D_{k-1}, G_{k-1}\rangle$ that contains words shorter than k. To guarantee completeness, all possible words of length k that appear in $\mathcal{S}$ are tested. The algorithm scans $\mathcal{S}$ once, tabulates, using a hashing scheme, all exact words of length k in $\mathcal{S}$, and computes their over-representation. The words that are over-represented are further classified as motif words or background words.

A word is considered over-represented if it occurs more frequently in $\mathcal{S}$ than it can be generated by the model $\langle D_{k-1}, G_{k-1}\rangle$. We measure the over-representation by a Z-score. Let N_w be the number of occurrences of a word w in $\mathcal{S}$ and random variable $\hat{N}_w$ the number of occurrences of w in the sequences with the same length as $\mathcal{S}$ which were supposedly generated by model $\langle D_{k-1}, G_{k-1}\rangle$. Denote $E(\hat{N}_w)$ and $\sigma(\hat{N}_w)$ as the mean and standard deviation of $\hat{N}_w$. The Z-score of w is defined as $Z_w = (N_w - E(\hat{N}_w))/\sigma(\hat{N}_w)$.

It is nontrivial to efficiently compute or estimate the statistics of random variable $\hat{N}_w$ and thus the Z-score of word w. The Z-score of a word in a random string generated by a Markovian process has been studied if all the words has a fixed length (Regnier, 1998). However, the words in our model may be of different lengths, and thus the computation of its statistics is more technically involved. Consider again a word w of length k in a sequence of length L generated by model $\langle D_{k-1}, G_{k-1}\rangle$, where $G_{k-1} = (\Psi, \Theta, \mathcal{I})$ was optimized in the model optimization component. There are various ways to produce w, for example, by concatenating words of single bases, or by merging a word's suffix with another word's prefix, etc. The expected number of occurrences, $\hat{N}_w$, should take all possible situations into account. In the case where w can only be exact concatenations of the words in D_{k-1}, $\hat{N}_w$ can be computed as $(L-k+1)P(w|\langle D_{k-1}, G_{k-1}\rangle)$. We define $\mathcal{A}_w(i)$ (and respectively $\mathcal{B}_w(j)$) to be the set of motifs in D_{k-1} whose suffixes (and respectively prefixes) match the first i (and respectively the last j) letters of w, then the expectation $E(\hat{N}_w)$ can be computed as,

$$\begin{aligned} E(\hat{N}_w) = \quad & (L-k+1) \cdot \left(\sum_{i=1}^{k} A_w(i) \cdot \right. \\ & \left. \left(\sum_{j=i+1}^{k} P\left(w_{[i+1,j-1]}|\langle D_{k-1}, G_{k-1}\rangle\right) B_w(j)\right)\right), \end{aligned}$$

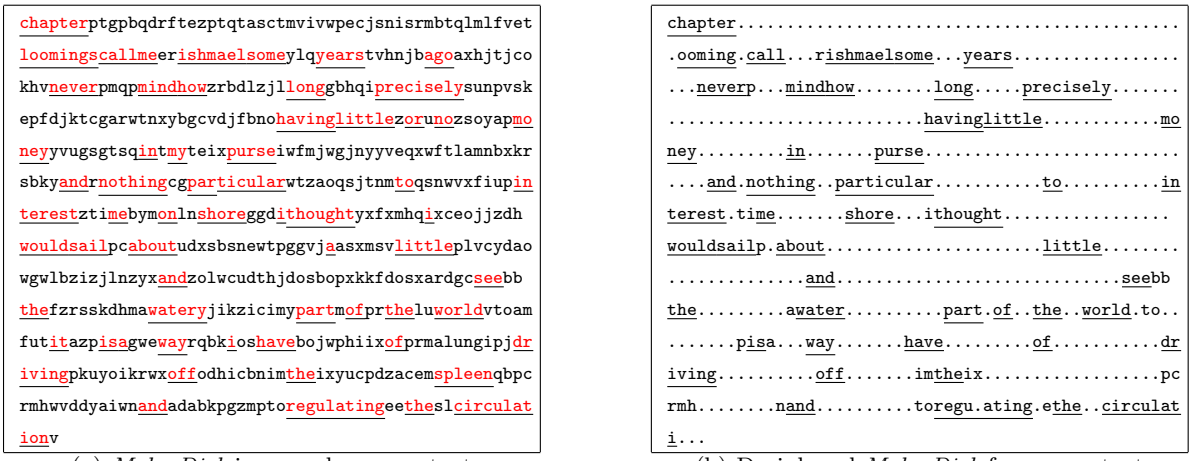
where

$$\begin{cases} A_w(i) = \sum_{W_u \in \mathcal{A}_w(i)} P_{W_u} P\left(w_{[1,i]}|\Theta_u^{(suffix_i)}\right), \\ B_w(j) = \sum_{W_u \in \mathcal{B}_w(j)} P_{W_u} P\left(w_{[j,k]}|\Theta_u^{(prefix_j)}\right), \end{cases}$$

and P_{W_u} is the transition probability of motif W_u in G_{k-1}, $\Theta_u^{(suffix_i)}$ and $\Theta_u^{(prefix_j)}$ the emission probabilities of the last i and last j positions of Θ_u, respectively. The computation of $\sigma(\hat{N}_w)$ is much complex and costly. In our current implementation, we used $E(\hat{N}_w)$ to approximate $\sigma(\hat{N}_w)$.

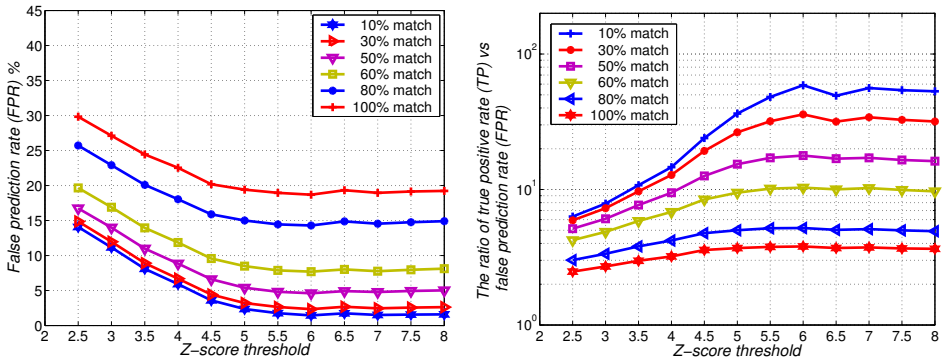
4. Decipher a stegoscript in English

We applied WordSpy to a stegoscript (about 268K letters) that had the first ten chapters (about 112K letters) of novel *Moby Dick* embedded within. This stegoscript was created by Bussemaker et al. (Bussemaker et al., 2002). Fig. 2(a) shows a tiny portion of the stegoscript, where the underlined text is the title and first two sentences of Chapter One. We ran WordSpy with different Z-score thresholds and to find words of maximum length of 15. We measured performance by the true positive rate (TPR), the percentage of true words discovered over all the words in the original text, and false prediction rate (FPR), the percentage of false predictions in the deciphered text. In measuring these rates, we considered different degrees of matches between a word in the original text and a predicted word



(a) *Moby Dick* in a random covertext (b) Deciphered *Moby Dick* from covertext

Figure 2. Deciphering novel *Moby Dick* from a stegoscript. (a) A small portion of the script; the underlined text is the title and first two sentences of Chapter One. (b) Deciphered *Moby Dick*. The identified background words are marked out by dots.



(a) (b)

Figure 3. Evaluation of WordSpy on a stegoscript of *Moby Dick*. (a) False prediction rates on different Z-score thresholds. (b) The ratios of true prediction rate over false prediction rate on different Z-score thresholds. The results are listed for different word matching ratios.

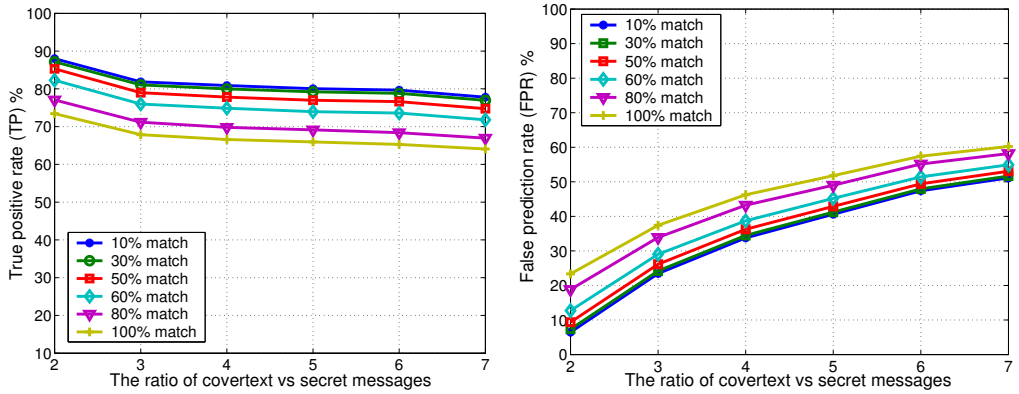
in the deciphered text. For example, we consider it as a correct prediction if a recovered word has at least a certain percent of letters matched to the original word, which we call word match rate. As summarized in Table 1, TPR decreases and FPR increases as the word match rate increases. If we take the most stringent criterion of 100% word match rate, WordSpy is able to recover ∼70% *exact* original words with a false prediction rate of ∼19% using Z-score threshold 6. As a comparison, when the word match rate is 50%, TPR increases to ∼82% while FPR decreases to ∼4.6%.

A close examination showed that the FPR initially decreases and then stays relatively constant as the Z-score threshold increases (Fig.3(a)). When the Z-score threshold is high enough (>5.5), most falsely predicted words will be filtered out. On the other hand, the true positive rate (TPR) always decreases as the Z-score threshold increases. The overall best performance seems to be reached around the Z-score threshold of 6 (Fig.3(b)).

To analyze the complexity of the problem, we test the algorithm on the some stegoscripts with different size of covertext. Using the same article *Moby Dick*, we generate 6 scripts with different covertext to secret messages ratio, from 2 to 7. We expect that as the size of covertext becomes larger, the deciphering problem becomes harder. Fig. 4 shows the results with Z-score threshold 3 on all 6 scripts. As expected, the true positive rate goes lower as the covertext becomes larger, while the false prediction rate goes higher at

Word match ratio (%):	10	20	30	40	50	60	70	80	90	100
True words discovered:	16047	16026	15899	15670	15529	15046	14584	14066	13500	13435
True positive rate (%):	84.7	84.6	83.9	82.7	82.0	79.4	77.0	74.3	71.3	70.9
False words reported:	238	259	387	617	761	1272	1787	2361	3003	3087
False prediction rate (%):	1.4	1.5	2.3	3.7	4.6	7.6	10.8	14.2	18.1	18.6

Table 1. Results on a stegoscript containing the first ten chapters of novel *Moby Dick* for Z-score threshold 6. Total 18930 words are in the original text. Total discovered words in the deciphered text are 16522. *Word match ratio* determines the least percentage of position matches for a true word to be considered correctly predicted. *True words discovered* gives the numbers of true words correctly predicted. *True positive rate* is the percentage of *true words discovered* over the total words in the original text. *False words reported* is the number of words falsely predicted based on different word match ratios. *False prediction rate* is the percentage of false words reported over the total words in the deciphered text.



(a) (b)

Figure 4. Prediction results on stegoscripts with different size of covertext. (a) True positive rate on different size of covertext. (b) False prediction rate on different size of covertext.

the same time. However, even as the covertext is as 7 times big as the original article, WordSpy still did a decent job. It accurately predicted 75% of the article and only had 53% of its predictions false positive, i.e., one will be correct for each two predictions.

To complete our example in Fig. 2(a), we show the recovered text in Fig. 2(b), where the identified background words are marked out with dots. This simple example interestingly shows that the deciphered text is pretty much readable.

5. Identify TFBMs of yeast cell-cycle genes

WordSpy is capable of genome-wide motif finding. We applied it to discovering TFBMs of ∼800 cell-cycle related genes of *S. cerevisiae* (Spellman et al., 1998). The promoter sequences are retrieved using the RSA tools (van Helden et al., 2000). We compared WordSpy with the MobyDick algorithm (Bussemaker et al., 2002), finding motifs with lengths upto 12. We tuned MobyDick to get its best possible parameters. The Z-score threshold for WordSpy was set to 3. Table 2 lists the results. As shown, WordSpy is able to identify all known cell-cycle TFBMs with high significant levels (Z-scores), and all of them (except one) are in the exact lengths. In contrast, MobyDick failed to discover four of them.

However, many known motifs discovered by WordSpy are not ranked very high using Z-score in our dictionary. This implies that Z-score alone is not sufficient to discern true TFBMs from background words. To identify truly biologically meaningful motifs, we need additional information.

Co-regulated genes very often tend to have similar expression profiles over different conditions. We can thus evaluate the likelihood of a motif being biologically meaningful by the coherence of the expression profiles of all the genes whose promoters contain the motif. We use the average coherence of pairwise gene expression profiles to measure the coherence of a set of expression profiles, and call this measure *G-score*, where *G* stands for genes. Therefore, a higher G-score indicates a more biologically meaningful motif.

We applied this G-score to order the motifs discovered by WordSpy for the yeast cell-cycle genes. We used the yeast gene expression data from (Stuart et al., 2003).

Known motifs	Known TFs	WordSpy	G-score rank	Z-score	Z-score rank	MobyDick
TGCTGG(CCAGCA)	Ace2,Swi5	TGCTGG	22	5.5	106/147	TGCTGCTGGA
RRCCAGCR(YGCTGGYY)	Ace2,Swi5	GCTGG	10	5.3	17/30	TGCTGCTGGA
ACGCGT(ACGCGT)	Swi6, Mbp1	ACGCGT	1	17.4	19/147	AACGCGT
CACGAAA(TTTCGTG)	Swi4, Swi6	CACGAAA	47	5.4	246/ 419	GTCACGAAA
CGCGAAA(TTTCGCG)	Swi4, Swi6	CGCGAAA	8	16.1	24/419	CGCGAAA
ATAAACAA(TTGTTTAT)	Fkh1,Fkh2	ATAAACAA	44	8.8	193/1015	TTGTTTAT
GTAAACAA(TTGTTTAC)	Fkh1,Fkh2	GTAAACAA	21	8.5	202/1015	n/a
GTAAACA(TGTTTAC)	Fkh1,Fkh2,Ndd1	GTAAACA	21	7.6	128/419	GTAAACA
TTTCCTAA(TTAGGAAA)	MCM1	TTTCCTAA	28	6.4	359/1015	n/a
TCACGTG(CACGTGA)	Met4,Met28,Cbf1	TCACGTG	93	5.0	288/419	n/a
TGAAACA(TGTTTCA)	Ste12	TGAAACAA	55	5.5	489/1015	n/a

Table 2. Discovered known motifs in the ∼800 promoters of yeast cell-cycle genes. The first two columns list the known motifs (and their reverse complimentary) and their potential TFs. The next four columns report the results from WordSpy, followed by the last column for MobyDick. The *Z-score rank* is based on motif Z-score, where the first number is the ranking and the second is the total number of discovered motifs of the same length. The *G-score rank* is the ranking among the motifs of the same length.

Interestingly, most known motifs are now ranked high in our dictionary (*G-score rank* in Table 2). Using the G-score, background words can be more accurately identified, and consequently a more accurate statistical grammar can be constructed to model the promoter sequences. This encouraging result suggests that the motifs in our dictionary which have high G-score rankings have good chances to be real motifs.

6. Conclusions and Discussion

In this research, we viewed a genome as a stego script with conserved functional sequence units as secret messages embedded in a covertext of "garbage" sequences. We also hypothesized that such a stego genomic script was constructed based on a dictionary and a grammar. We then proposed and developed an approach to discover the words in the dictionary and learn the grammar. Specifically, we used stochastic dictionary, which allows degenerate words or motifs, and stochastic regular grammar. We also considered how to recover the original messages in this paper.

We applied our algorithm, called WordSpy, to two sets of large real text data. We used WordSpy to recover the first ten chapters of novel *Moby Dick*, which was intentionally embedded into a random text. WordSpy is able to recover the original text with accuracy more than 80% which is very surprising considering that some words in the original text, such as the words with single copy, are stochastically unidentifiable. We also applied WordSpy to identify the transcription factor binding motifs (TFBMs) in the upstream regulatory regions of cell-cycle related genes in budding yeast *S. cerevisiae*. WordSpy significantly outperforms existing algorithm MobyDick. WordSpy found all cell-cycle related TFBMs with high significance, while the existing algorithm missed many of them.

Our approach and algorithm can be applied to many problems in many different ways. For our motivating application, our method can be used to finding TFBMs in non-coding regions of a genome. We are also interested in applying our method to decipher unknown and unfamiliar scripts in natural language domain.

References

Bailey, T., & Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using EM. *Machine Learning*, *21*, 51–80.

Bailey, T., & Noble, W. (2003). Searching for statistically significant regulatory modules. *Bioinformatics*, *19*, 16–25.

Blaiseau, P., & Thomas, D. (1998). Multiple transcriptional activation complexes tether the yeast activator Met4 to DNA. *EMBO J*, *17*, 6327–36.

Brivanlou, A., & Darnell, J. (2002). Signal transduction and the control of gene expression. *Science*, 813–8.

Bussemaker, H., Li, H., & Siggia, E. (2002). Building a dictionary for genomes: Identification of presumptive regulatory sites by statistical analysis. *Proc. Natl. Acad. Sci. USA.*, *97*, 10096–100.

Cliften, P., Sudarsanam, P., Desikan, A., Fulton, L., Fulton, B., Majors, J., Waterston, R., Cohen, B., & Johnston, M. (2003). Finding functional features in Saccharomyces genomes by phylogenetic footprinting. *Science*, *301*, 71–6.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J Royal Statistical Society*, *39*, 1–38.

Dolan, J., Kirkman, C., & Fields, S. (1989). The yeast STE12 protein binds to the DNA sequence mediating pheromone induction. *Proc. Natl. Acad. Sci. USA.*, *86*, 5703–7.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* Cambridge University Press.

Frith, M., Hansen, U., & Weng, Z. (2001). Detection of cis-element clusters in higher eukaryotic dna. *Bioinformatics*, *17*, 878–889.

Hertz, G., & Stormo, G. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, *15*, 563–77.

Hughes, J., Estep, P., Tavazoie, S., & Church, G. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Molecular Biology*, *296*, 1205–14.

Kato, M., Hata, N., Banerjee, N., Futcher, B., & Zhang, M. (2004). Identifying combinatorial regulation of transcription factors and binding motifs. *Genome Biology*, *5*, R56.

Knight, K., & Yamada, K. (1999). A computational approach to deciphering unknown scripts. *Proc ACL Workshop on Unsupervised Learning in Natural Language Processing*.

Lawrence, C., Altschul, S., Bogouski, M., Liu, J., Neuwald, A., & Wooten, J. (1993). Detecting subtle sequence signals: A gibbs sampling strategy for multiple alignment. *Science*, *262*, 208–14.

Lemon, B., & Tjian, R. (2000). Orchestrated response: A symphony of transcription factors for gene control. *Genes Dev.*, *14*, 2551–69.

Regnier, M. (1998). Unified approach to word statistics. *RECOMB* (pp. 207–13).

Rivest, R. (1998). Chaffing and winnowing: Confidentiality without encryption. *CryptoBytes*, *4*, 12–7.

Searls, D. (1992). The linguistics of DNA. *American Scientist*, *80*, 579–91.

Sinha, S., Nimwegen, E., & Siggia, E. (2003). A probabilistic method to detect regulatory modules. *Bioinformatics*, *19*, 292–301.

Sinha, S., & Tompa, M. (2000). A statistical method for finding transcription factor binding sites. *8th Intern. Conf. on Intelligent Systems for Molecular Biology* (pp. 344–54).

Spellman, P., Zhang, M., Lyer, V., Anders, K., Eisen, M., abd D. Botstein, P. B., & Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, *9*, 3273–97.

Stuart, J., Segal, E., Koller, D., & Kim, S. (2003). A gene coexpression network for global discovery of conserved genetic modules. *Science*, *302*, 249–55.

USA TODAY (2001). Terror groups hide behind web encryption. http://www.usatoday.com/tech/news/2001-02-05-binladen.htm.

van Helden, J., Andre, B., & Collado-Vides, J. (1998). Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Molecular Biology*, *281*, 827–42.

van Helden, J., Andre, B., & Collado-Vides, J. (2000). A web site for the computational analysis of yeast regulatory sequences. *Yeast*, *16*, 177–87.

Wayner, P. (2002). *Disappearing Cryptography*. Morgan Kaufmann. 2 edition.

Xing, E., & Wu, W. (2003). LOGOS: a modular Bayesian model for de novo motif detection. *Proc. Computational Systems Bioinformatics (CSB'03)*, 266–276.

Zhu, J., & Zhang, M. (1999). SCPD: A Promoter Database of Yeast Saccharomyces cerevisiae. *Bioinformatics*, *15*, 607–11.