ENTROPIA DO DNA

JOSÉ OSVALDO COUTO HORTA

RESUMO. Neste trabalho descrevemos as atividades realizadas para disciplina MAC-5701 Estudo Dirigido. O nosso objetivo foi compreender algumas aplicações da Teoria da Informação à Bilogia Molecular, mais especificamente, estudamos técnicas para a estimativa da entropia da estrutura primária do DNA.

1. Introdução

O mundo científico vem se dedicando intensamente ao estudo do DNA e ultimamente ao seu seqüenciamento. A representação da estrutura primária do DNA geralmente é muito extensa o que leva à utilização de computadores. O DNA contém em si toda a informação necessária para o desenvolvimento e funcionamento dos seres vivos. A entropia é um dos parâmetros utilizados para medir a quantidade de informação. A Entropia do DNA tem sido alvo de acurados estudos (cf. [6, 5, 7]). Este é um tema complexo que exige tanto a compreensão da teoria da informação quanto a dos conceitos biológicos envolvidos. O estudo da Entropia do DNA prioriza a medição da complexidade e conteúdo da informação em detrimento de uma elevada taxa de compressão.

A compressão também está ligada à previsibilidade. A previsibilidade de alguma porção do DNA pode possuir um significado biológico. A baixa taxa de compressão da representação do DNA indica a possibilidade da existência de componentes aleatórios na formação (por evolução) do DNA. A análise da Entropia do DNA pode ajudar na localização de suas porções que possuam significado biológico. Espécies que apresentam porções do DNA parecidas podem ser parentes ou mesmo possuírem características comuns. O estudo da entropia pode permitir a análise de forma eficiente da previsibilidade do DNA ou de alguma porção do mesmo, permite também medir de forma quantitativa as informações nele contidas, passando dados para uma averiguação do significado biológico dos dados obtidos.

Os genes são porções do DNA e são a base da genética. Mesmo as formas de vida mais primitivas os possuem. Um dos feitos científico mais importantes deste final de milênio foi a codificação completa do DNA de um ser humano. É necessário agora identificar cada porção do DNA, para que seja possível o estudo não só de seu significado mas também a identificação de suas funções biológicas. A Entropia do DNA é uma ferramenta poderosa que pode auxiliar bastante na efetiva identificação dos genes humanos.

Date: 30 de agosto de 2000.

Neste trabalho a proposta principal está no estudo da Entropia do DNA. O que se sabe previamente é que a estimativa da Entropia do DNA não é um problema simples.

Para se conseguir caracterizar este problema e entende-lo melhor foi feita a quebra do problema em unidades mais simples.

Este trabalho foi organizado da seguinte maneira.

Na Seção 2, discutimos sobre os princípios básicos da biologia molecular que foi extraída do livro *Biologia Hoje Volume I* [3]. Os conceitos que necessitavam de um enfoque mais voltado à teoria da informação foram extraídos do livro *Algorithms on Strings, Trees, and Sequences Computer Science and Computational Biology* [4] e do artigo *Protein is Incompressible* [7].

Na Seção 3 tratamos dos princípios básicos da teoria da informação, que por estar muito ligada à compressão de textos foi estudada basicamente pelo livro *Text Compression* [1] e complementada pelos livros *Introduction to Data Compression* [8] e *Algorithms on Strings, Trees, and Sequences Computer Science and Computational Biology* [4]. Nesta unidade a proposta principal foi buscar na teoria da informação a base para se chegar ao conceito de entropia.

O conceito de modelo é fundamental para a estimativa da entropia. Na Seção 4 veremos os diferentes tipos de modelo que foram estudados nos livros *Text Compression* [1] e *Introduction to Data Compression* [8].

O conhecimento da informação contida na linguagem natural possui não apenas um aspecto histórico mas também serviu de base para a maioria dos algoritmos de compressão existentes. Na Seção 5 falamos a este respeito. Os estudos feitos sobre a Entropia do DNA de alguma forma remetem aos algoritmos desenvolvidos para a linguagem natural, como princípio para a criação do algoritmo utilizado ou como referência para comparações. O livro *Text Compression* [1] foi a base desta unidade.

Na Seção 6 descrevemos algumas das principais técnicas da compressão de dados. Dedicamos bastante tempo a esta unidade pois é o estudo destas técnicas que permite compreender e avaliar os diversos estudos a respeito da Entropia do DNA. Esta seção acabou recebendo dois apêndices para complementar e auxiliar a compreenção do texto principal. Nesta seção utilizamos os livros *Text Compression* [1] e Introduction to Data Compression [8] como base e completamos alguns tópicos com o livro *Algorithms on Strings, Trees, and Sequences Computer Science and Computational Biology* [4]. Embora nesta seção haja alguma referência à estruturas de dados que sustentam os algoritmos de compressão não é este o nosso assunto. Tais referências são vagas e servem somente para aprimorar o contexto de um ou outro tópico.

Após o estudo da compressão de dados passamos para os artigos científicos. NA Seção 7 analisamos quatro artigos a respeito da Entropia do DNA. Os artigos *Significantly Lower Entropy Estimates for Natural DNA Sequences* [6]; *Protein is Incompressible* [7]; *Estimanting DNA sequence Entropy* [5] e *On the Entropy of DNA: Algorithms and Measurements based on Memory and Rapid Convergence* [2] nos mostram os principais aspectos envolvidos na Entropia do DNA. Abordamos os artigos fazendo um resumo de cada trabalho desenvolvido em cada artigo e analisando as suas principais conclusões. Além disto colocamos alguns gráficos e tabelas que destacam alguns resultados dos artigos para facilitar a análise dos resultados obtidos pelos autores destes trabalhos.

Ao fazermos a conclusão deste trabalho, na Seção 8, interpretamos o problema da Entropia do DNA e os principais obstáculos para seu cálculo.

2. Conceitos Básicos da Biologia Molecular

O DNA, abreviação em inglês para ácido desoxirribonucléico (ADN em português), foi descoberto em 1869 pelo bioquímico alemão Johann Miesher (1844-1895). O DNA é composto por nucleotídeos. Os *nucleotídeos* são compostos por três tipos de substâncias químicas: bases nitrogenadas (composto cíclico contendo nitrogênio), pentose (açúcar de cinco átomos de carbono) e fosfato (radical do ácido fosfórico). Na região do núcleo das células os nucleotídeos se reúnem formando longos filamentos chamados *polinucleotídeos*. A união dos nucleotídeos é sempre feita entre o fosfato de um nucleotídeo com a pentose do nucleotídeo vizinho. Há 5 tipos principais de bases nitrogenadas: adenina; guanina; citosina; timina; e uracil. As bases adenina e guanina são conhecidas como bases *purínicas* ou *púricas* derivadas da purina. As outras três bases são derivadas de um outro composto chamado pirimidina e são chamadas de bases *pirimidínicas* ou bases *pirimídicas*. As pentoses são de dois tipos, ribose ou desoxirribose. Existem dois tipos de ácidos nucléicos: o ácido desoxirribonucléico (DNA ou ADN) e o ácido ribonucléico (RNA ou ARN).

Em 1953 James Watson e Francis Crick apresentaram a estrutura do DNA. Segundo o modelo de Watson e Crick a molécula do DNA é constituída por duas cadeias polinucleotídicas dispostas em hélice ao redor de um eixo imaginário. As cadeias polinucleotídicas ficariam unidas por meio de pontes de hidrogênio que são mantidas entre bases específicas, a adenina com a timina e a citosina com a guanina. Estas duas cadeias formam uma estrutura em espiral, de dupla hélice, girando para a direita. A base uracil substitui a timina no RNA, já que no RNA não há timina e no DNA não há uracil.

As cadeias do DNA são complementares entre si pois devido à estreita forma de ligação nas moléculas do DNA se em uma determinada posição de uma das duas cadeias de polinucleotídeos houver uma adenina na posição equivalente da outra cadeia haverá necessariamente uma timina e vice-versa. O fato de as cadeias do DNA serem complementares simplificou a forma de representar a molécula de DNA, pois representando somente uma das cadeias indiretamente se está representando também a outra cadeia.

Para simplificar a representação do DNA convencionou-se a utilização das letras `‘a’`, `‘c’`, `‘g’`, `‘t’`, `‘u’` para representar as bases adenina, citosina, guanina, timina e uracil, respectivamente. Embora haja uma ordem rígida para a ligação entre as duas cadeias polinucleotídicas o mesmo não ocorre na ligação entre as bases de uma mesma cadeia.

Dentre as principais funções do DNA está a produção de proteínas. Para a compreensão deste processo há vários conceitos que devem estar claros. *Procariotes* são seres vivos elementares que não possuem seu DNA restrito ao núcleo. Já *eucariotes* são seres superiores que além de restringirem o DNA ao núcleo das células possuem seus genes divididos em duas regiões que se intercalam sucessivas vezes, os chamados *introns* e *exons*. *Os genes* são as regiões contíguas do DNA que contêm as informações necessárias para a produção das proteínas, são também as unidades básicas da hereditariedade. Os *introns* são as regiões dos genes que não estão ligadas à produção de proteínas e suas funções ainda não são conhecidas. Os *exons* são as regiões dos genes que contêm as informações que são utilizadas pelo RNA mensageiro e que são utilizadas na produção das proteínas e em alguns casos na produção do RNA.

Em um organismo complexo, como o organismo humano, as células são especializadas e não é toda célula que produz cada tipo de proteína. O processo proposto pelos cientistas franceses François Jacob, André Lwoff e Jacques Monod (Prêmio Nobel de 1965) prevê a existência de quatro tipos de genes ao longo da molécula de DNA, o gene estrutural, o gene regulador, o gene promotor e o gene operador. Os genes estruturais são responsáveis pela síntese de moléculas do RNA mensageiro. O RNA mensageiro é o responsável pela síntese de proteínas. O gene promotor somente pode se ligar ao RNA-polimerase se seu vizinho gene operador estiver livre. Caso o gene operador esteja inibido o gene promotor também estará e como gene operador não se ligará ao RNA-polimerase o processo de síntese do RNA mensageiro não será ativado no gene estrutural. Toda esta ativação está ligada à proteína repressor que é produzida pelo gene regulador. Desta forma, de acordo com os genes ativados ou desativados em cada célula a célula exerce uma função específica de seu tipo.

As proteínas são cadeias polinucleotídicas produzidas pelo RNA mensageiro a partir de um segmento contíguo do DNA. Porém ao contrário do DNA nas proteínas a principal referência não está nos nucleotídeos mas nos aminoácidos. Os aminoácidos são pequenas moléculas que servem de base para a sintetização das proteínas. Há na natureza exatamente 20 tipos de aminoácidos. Cada aminoácido é formado por exatamente três bases nitrogenadas. Os códons são seqüências de três bases nitrogenadas. Dos 64 possíveis códons distintos três deles não representam aminoácidos os demais representam cada um dos 20 aminoácidos existentes.

Código	aminoácido	trios de bases (códons) no DNA
A	Alanina	gcu, gcc, gca, gcg
C	Cisteína	ugu, ugc
D	Ácido Aspártico	gau, gac
E	Ácido Glutâmico	gaa, gag
F	Fenilalanina	uuu, uuc
G	Glicina	ggu, ggc, gga, ggg
H	Histidina	cau, cac
I	Isoleucina	auu, auc, aua
K	Lisina	aaa, aag
L	Leucina	uua, uug, cuu, cuc, cua, cug
M	Metionina	aug
N	Asparagina	aau, aac
P	Prolina	ccu, ccc, cca, ccg
Q	Glutamina	caa, cag
R	Arginina	cgu, cgc, cga, cgg, aga, agg
S	Serina	agu, agc, ucu, ucc, uca, ucg
T	Teonina	acu, acc, aca, acg
V	Valina	guu, guc, gua, gug
W	Triptófano	ugg
Y	Tirosina	uau e uac

Os códons uaa, uga e uag não representam nenhum aminoácido em específico e sim representam o final de um intron e a Metionina também representa o início de uma região exon [3]. Podemos destacar que embora hajam apenas 20 aminoácidos há 61 códons que sintetizam estes aminoácidos, há vários aminoácidos com mais de uma forma de sintetização.

A Biologia Molecular é uma ciência complexa e as definições acima servem como um guia rápido para que seja possível a compreensão dos artigos que tratam da Entropia do DNA, sem ter a pretensão de ser completo ou abrangente. Caso algum novo conceito seja necessário este será inserido no local mais adequado.

3. Conceitos Básicos da Teoria da Informação

Alguns conceitos básicos são indispensáveis ao estudo da Entropia do DNA. *Compressão* é a tentativa de reescrever a mensagem utilizando menos espaço. *Mensagem* é o alvo da compressão. O processo de comprimir também é chamado de *codificação*, já o processo inverso de *decodificação*. O *transmissor* é aquele que efetiva a codificação e o *receptor* o que faz a decodificação. Uma mensagem é representada através de uma seqüência de finita sobre um *alfabeto*, conjunto finito de códigos possíveis de aparecer em uma mensagem. A mensagem guarda em si uma certa quantidade de informação que não está diretamente relacionada ao tamanho da mensagem. Um exemplo está na representação dos números. Se quisermos representar todos os números naturais de 1 a 1.000.000 podemos utilizar

$$(1\times 7)+(900.000\times 6)+(90.000\times 5)+(9.000\times 4)+(900\times 3)+(90\times 2)+(9\times 1)$$

algarismos (ao representar cada número) = 5.888.896 algarismos ou 42 caracteres da mensagem "todos os números naturais de 1 a 1.000.000"ou mais compactado ainda a expressão matemática: $1, 2, \ldots, 1.000.000$ com exatos 17 caracteres.

A previsibilidade de uma mensagem interfere diretamente na eficiência de sua compressão. Quanto mais previsível for a mensagem melhor será a sua taxa de compressão. O *modelo* é um meio utilizado para definir a distribuição estatística de uma mensagem bem como sua previsibilidade. O modelo interfere diretamente na previsibilidade de uma mensagem.

Consideremos o seguinte exemplo. Dado um alfabeto $A = \{a, b, c, d, e\}$ e a mensagem $M = eaebbdecbaedbdecaaedbaecddecbadd$. Seja M_1 o modelo aplicado pelo processo de contar as ocorrências de cada símbolo do alfabeto e obter a probabilidade de cada símbolo pela divisão do contador do símbolo pelo tamanho da mensagem. Para M_1 a distribuição de probabilidades seria

- $p(a) = 6/32$;$p(b) = 6/32$; $p(c) = 4/32$; $p(d) = 8/32$; $p(e) = 8/32$.

Seja M_2 o modelo aplicado pelo processo de se considerar que existem quatro estados vinculados à mensagem M: O primeiro estado aplicado aos símbolos que ocupam as posições $1, 5, 9, 13, \ldots$ de M. O segundo estado aplicado aos símbolos que ocupam as posições $2, 6, 10, 14, \ldots$ de M. O terceiro estado aplicado aos símbolos que ocupam as posições $3, 7, 11, 15, \ldots$ de M e o quarto estado aplicado aos símbolos que ocupam as posições $4, 8, 12, 16, \ldots$ de M. Como o tamanho de M é de 32 cada estado possui, neste exemplo, exatamente 8 símbolos. Utilizaremos a notação $p(i, s)$ para denotar a probabilidade do símbolo s no estado i. Neste caso a distribuição das probabilidades para cada símbolo seria

- $p(1, a) = 1/8$; $p(1, b) = 5/8$; $p(1, c) = 0/8$; $p(1, d) = 1/8$; $p(1, e) = 1/8$, para o primeiro estado;
- $p(2, a) = 5/8$; $p(2, b) = 0/8$; $p(2, c) = 0/8$; $p(2, d) = 3/8$; $p(2, e) = 0/8$, para o segundo estado;
- $p(3, a) = 0/8$; $p(3, b) = 0/8$; $p(3, c) = 0/8$; $p(3, d) = 1/8$; $p(3, e) = 7/8$, para o terceiro estado;
- $p(4, a) = 0/8$; $p(4, b) = 1/8$; $p(4, c) = 4/8$; $p(4, d) = 3/8$; $p(4, e) = 0/8$, para o quarto estado.

O modelo M_2 forneceu uma melhor previsibilidade que M_1 pois apresentou uma distribuição de probabilidades menos uniforme.

O processo de compressão pode ser reversível ou irreversível. A compressão *reversível* (sem perdas) é aquela que permite recuperar a mensagem original integralmente, é o tipo de compressão normalmente utilizado para textos. A compressão *irreversível* é aquela que não permite a recuperação integral da mensagem original, um exemplo é a conversa telefônica. Na telefonia há pequenas alterações na voz dos interlocutores, mas é perfeitamente possível distinguir quem está falando e o que está sendo dito. A compressão depende da quantidade de informação. A quantidade de informação depende da previsibilidade da mensagem. Quanto maior for a previsibilidade da mensagem menor será a quantidade de informação contida na mensagem. Já a previsibilidade da mensagem depende do modelo utilizado para gerar a distribuição estatística. Quanto menos uniforme for a distribuição estatística mais previsível é a mensagem.

A *entropia* é a medida da quantidade de informação. Dada a distribuição de probabilidades $P = \{p_1, p_2, \ldots, p_n\}$ sendo $p_1 + p_2 + \cdots + p_n = 1$. A entropia E aplicada à P é:

$$E = -\sum_{i=1}^{n} p_i \log p_i \ bits.$$

O sinal menos serve para tornar posivita a entropia E.

Shannon mostrou que o melhor esquema de compressão sem perdas possui o número de bits na média igual à entropia da fonte (mensagem), para o modelo adotado.

Mais a frente será feita um ligeiro aprofundamento na teoria da modelagem e nas técnicas de compressão mais utilizadas na compressão de textos.

É necessário algum conhecimento básico da biologia molecular pois não é possível abordar a Entropia do DNA sem se conhecer os conceitos básicos que envolvem o DNA em si uma vez que o modelo utilizado para definir a previsibilidade do DNA depende mais das características dele que da correta utilização da teoria da informação.

4. Modelos e Compressão

Há duas formas distintas de lidarmos com os modelos. A primeira se refere à forma de definir o modelo a segunda sobre os tipos de modelos.

Os modelos podem ser estáticos, semi-adaptativos ou adaptativos. Os modelos estáticos são conhecidos previamente pelo transmissor e receptor de forma que tanto a codificação quanto a decodificação independem da mensagem. Esta forma de definir o modelo tem por principal vantagem que não há perda de tempo no processo de manutenção do modelo. No modelo semi-adaptativo o codificador lê toda a mensagem e cria um livro de códigos com as palavras mais utilizadas transmitindo para o receptor o livro de código e em seguida a mensagem. Neste caso há um custo para a manutenção do modelo (livro código) mas há uma melhor taxa de compressão uma vez que os códigos do livro refletem a mensagem para o qual foi criado. A terceira forma de definir o modelo é a forma adaptativa. Nesta forma, como nos modelos estáticos, há um modelo inicial que existe tanto no transmissor quanto no receptor e a medida que o código vai sendo transmitido tanto o receptor quanto o transmissor alternam da mesma forma o modelo. Nos modelos adaptativos há um custo para se manter o modelo e normalmente há a necessidade de uma estrutura complexa para atender de forma eficiente às alterações dinâmicas do modelo.

4.1. **Modelo de contexto finito.** O *modelo de contexto finito* é um tipo de modelo que parte do conceito que a probabilidade do próximo símbolo será dada pela seqüência de alguns símbolos antecessores deste próximo. A ordem do modelo é o número de símbolos que são considerados para gerar a estatística do próximo símbolo. O número de possibilidades para um modelo varia exponencialmente, em função do tamanho do alfabeto, com a ordem do modelo.

Tomemos como exemplo o alfabeto $A = \{a, t\}$ e a mensagem

$$M = \texttt{attaattaaaaatattaatttatttttaatt}.$$

No modelo de ordem 0 há duas possibilidades: $p(a) = 15/32$ e $p(b) = 17/32$. Na ordem zero é feita uma simples contagem das ocorrências de cada símbolo do alfabeto, pois não há considerações a respeito de antecessores. O mesmo ocorre na ordem -1 na qual há uma distribuição equiprovável para todos os símbolos do alfabeto. A partir da ordem 1 começam as considerações sobre os símbolos antecessores.A distribuição das probabilidades nos modelos de ordem superior a 0 implicam uma consideração sobre o(s) antecessor(es) imediatos. Será adotada a notação $p(s)_e$ para indicar a probabilidade do símbolo s após a ocorrência do evento e. Para o exemplo acima na ordem 1 há dois possíveis eventos A e T. Para este exemplo
$p(A)_A = 8/15$; $p(T)_A = 7/15$; $p(A)_T = 6/17$ e $p(T)_T = 10/17$. Há duas possibilidades para tornar as probabilidades após a ocorrência do evento T unitárias. A primeira é diminuir em 1 o contador dos eventos para o evento T e a segunda é incluir no alfabeto um símbolo especial para o fim da mensagem. Esta diferença ocorre pois a mensagem termina com o símbolo T. Neste trabalho será adotada a primeira solução pois não há mais mensagem a ser decodificada após o último símbolo. Para este exemplo a distribuição de probabilidades para o modelo de ordem 1 é:

$$p(A)_A = 8/15;\ p(T)_A = 7/15;\ p(A)_T = 6/16 \text{ e } p(T)_T = 10/16.$$

No modelo de ordem 2 as distribuições de probabilidade são:

$$p(A)_{AA} = 4/8,\ p(T)_{AA} = 4/8;\ p(A)_{AT} = 1/6;\ p(T)_{AT} = 5/6;$$
$$p(A)_{TA} = 4/6;\ p(T)_{TA} = 2/6;\ p(A)_{TT} = 5/9;\ p(T)_{TT} = 4/9.$$

Já a distribuição de probabilidades para a ordem 3 possui 16 probabilidades e na ordem 4 há 32 probabilidades para apenas 2 símbolos no alfabeto! O tamanho da mensagem é 32. O essencial do modelo de contexto finito está no fato de a distribuição das probabilidades ser feita baseada em um número (ordem) de símbolos antecessores ao símbolo que se deseja prever.

4.2. **Modelos de estado finito.** Está baseado nas máquinas de estado finito. Nos modelos de estado finito a distribuição das probabilidades é dada em função do estado que a mensagem estará na colocação do próximo símbolo. O modelo M_2 do exemplo dado quando da definição de modelo no item 2 é um modelo de estado finito. Um exemplo de modelo de quatro estados é o motor a explosão que possui exatamente 4 tempos. Os modelos de estado finito podem ser utilizados para implementar os modelos de contexto finito, fazendo de cada contexto um estado possível, já a recíproca não é verdadeira pois nem todo estado pode ser representados por contextos.

4.3. **Modelos gramáticos.** São modelos que definem bem linguagens de computação como Cobol, Pascal, ou qualquer linguagem de sintaxe rígida e bem definida. Estes modelos não são bons para representar a linguagem natural pois embora exista uma sintaxe, ela é variável e cheia de exceções como frases sem verbos (Fogo!), frases sem sujeito (Faz frio.), frases incompletas (Eu gostaria, mas…), verbos irregulares como o verbo ir, frases dentro de frases (João, que é o homem forte da sala, tremeu de medo.), frases que completam frases (Ele queria viajar mas o dinheiro era pouco.), etc…

4.4. **Modelos ergódicos.** *Ergodicidade* é a propriedade de uma seqüência se tornar inteiramente representativa de todo o modelo se esta for suficientemente longa. O seguinte modelo não é ergódico, pois uma seqüência começada em c ou d jamais representaria $\{a, b, c, d\}$ por mais que ela crescesse.

$$\begin{array}{lcl} a & \rightarrow & b|c|a \\ b & \rightarrow & b|c|a \\ c & \rightarrow & c|d \\ d & \rightarrow & c|d \end{array}$$

Já o modelo abaixo é ergódico, pois se uma dada seqüência for suficientemente grande esta poderá representar inteiramente a, b, c, d, qualquer que seja a condição de início ou decisão tomada no meio da seqüência.

$$\begin{array}{lcl} a & \rightarrow & d \\ b & \rightarrow & d \\ c & \rightarrow & d \\ d & \rightarrow & a|b|c \end{array}$$

Os exemplos acima encontram-se ilustrados na Figura 1.

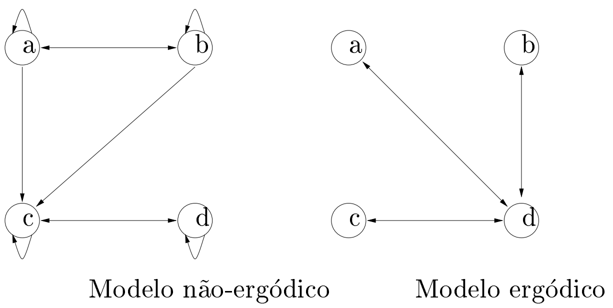


FIGURA 1. Exemplos de modelos ergódicos e não-ergódicos.

Todos os modelos de estado vinculados à linguagem natural são ergódicos. Um uso da ergocidade está no cálculo da entropia de uma mensagem. Se o modelo é ergódico ele tende a independer do ponto de início, já se o modelo não é ergódico o ponto de início da mensagem influencia bastante o cálculo da entropia. No primeiro modelo a, b, c, d acima suponha $p(a) = 0,6$; $p(b) = 0,3$; $p(c) = 0,08$ e $p(d) = 0,02$, uma mensagem qualquer iniciada por a ou b teria uma entropia muito diferente de uma mensagem iniciada por c ou d uma vez que não é possível para mensagens iniciadas por c ou d conter a ou b, apesar de $p(a) + p(b) = 0,9$! Já no segundo modelo $p(d) = 0,5$ se a mensagem for suficientemente grande. Para modelos ergódicos a entropia do modelo como um todo é a soma das entropias individuais de cada estado levando-se em conta o peso da probabilidade de cada estado.

No cálculo da entropia a ordem do modelo influencia diretamente na distribuição das probabilidades. As *probabilidades condicionais* são as probabilidades que um próximo símbolo ocorra caso um dado evento tenha ocorrido. Neste contexto *evento* é uma seqüência específica de símbolos. Em um contexto mais amplo a seqüência de símbolos remeteria ao início da mensagem, mas uma aproximação válida é calcular a probabilidade condicional para um número limitado de símbolos (dois símbolos por exemplo). Uma classe de condições é o conjunto das probabilidades condicionais para um dado evento. No estudo dos modelos adaptativos há algumas considerações que devem ser destacadas, para isto precisamos definir *envio* como sendo a transmissão da mensagem do codificador para o decodificador. Normalmente quando os modelos adaptativos são utilizados o envio ocorre símbolo a símbolo para que tanto o codificados quanto o decodificador possam atualizar suas estatísticas. Somente as probabilidades associadas a uma classe de condição e as próprias classes de condição podem ser alteradas nos processos adaptativos.

Nos modelos adaptativos há o problema da freqüência zero: Há um paradoxo clássico proposto por C. S. Pierce (1878): Dada uma urna com 1000 bolas diversas cores, dentre elas a branca e a preta. Supondo que tirada ao acaso uma bola da urna, se a pergunta for: A bola é preta? Há duas respostas possíveis, sim ou não, que pelo método adaptativo induziria a probabilidades de 50% para cada caso. Já se na mesma urna, ao se retirar uma bola ao acaso se a pergunta for é preta ou é branca ou nem preta e nem branca? Neste caso, como há três possíveis respostas associadas a probabilidade de a bola ser preta e 1/3, o que contradiz a probabilidade de 1/2 da pergunta anterior, para a mesma urna.

Outra característica importante dos modelos adaptativos é que: dado um conjunto de classes condicionais com K membros, um alfabeto de q símbolos e uma mensagem de tamanho n, há modelos adaptativos que serão no máximo $Kq\log_2 n$ bits maior que a melhor compressão possível em modelo não adaptativo. Por outro lado há um corolário deste resultado que um sistema adaptativo pode no máximo expandir uma mensagem para: $(q+1)/2 \times \log_2 n$ bits ao passo que um modelo estatístico pode expandir uma mensagem arbitrariamente.

Nem sempre é possível fazer uma estatística acurada da mensagem antes de comprimi-la e transmiti-la para o receptor (decodificador). Neste caso deve-se optar por se utilizar códigos adaptativos. O risco de se trabalhar com modelos determinísticos é grande, ao passo que nos modelos adaptativos as perdas, se ocorrerem, são limitadas. O importante neste caso é que tanto o codificador quanto o decodificador trabalhem com o mesmo modelo inicial, garantindo assim que a mensagem será corretamente decodificada.

O fato de não se poder ter uma pré formatação das mensagens induz à utilização de métodos adaptativos que por sua vez nos dá um custo computacional extra. Por outro lado quando há a possibilidade de fazer uma pré formatação deve-se transmitir para o decodificador esta pré formatação, o que também é um custo que dependendo do modelo e da ordem utilizados pode ser até mais alto que o custo computacional da utilização dos modelos adaptativos.

5. Modelagem da Linguagem Natural

A modelagem da linguagem natural é muito utilizada por diversos compressores e é muito comum no ambiente de informática. Embora não haja relações diretas entre a linguagem natural e o DNA, o desenvolvimento das técnicas de compressão passaram pela compressão da linguagem natural e as ferramentas vindas da linguagem natural são freqüentemente utilizadas nas técnicas de compressão para o DNA, porém sem a eficiência que marca a compressão da forma mais habitual de comunicação do ser humano.

A linguagem natural é composta de orações, que são formadas de palavras espaços em branco e sinais de pontuação. As palavras são escritas com letras e sinais como apóstrofe, hífens e acentuação. Para se comprimir textos em linguagem deve-se definir qual a unidade e a ordem do modelo a ser adotada. Os conceitos de letra e palavra são adaptados para atender a modelagem dos textos em linguagem natural. As letras são qualquer caracter válido. *Palavra* é uma seqüência de símbolos que não contenham o espaço em branco e suas variações (tabulação, parágrafo, quebra de linha, etc...). Porém o conceito de palavra não é unânime podendo possuir outras definições em outros estudos. Os *digramas* são um conjunto de duas letras ou palavras dependendo do contexto. Os *trigramas* e *tetragramas* são respectivamente um conjunto de três ou quatro letras ou palavras dependendo do contexto. Em se tratando de linguagem natural nem todos os digramas, trigramas e tetragramas ocorrem. Na língua portuguesa o digrama cc não ocorre, o mesmo ocorrendo para o trigrama nhm e o tetragrama arrc. Estudos feitos para a língua inglesa, num alfabeto de 94 letras, mostram que apenas 39% dos digramas possíveis; 3,6% dos trigramas possíveis e 0,2% dos tetragramas possíveis ocorrem. Criando-se textos aleatórios, a partir de estatísticas de digramas, trigramas, ..., 12-gramas se consegue chegar cada vez mais perto do inglês, porém completamente sem sentido.

George Zipf publicou em 1949 o livro Comportamento Humano e o Princípio do Menor Esforço, no qual o autor determina que as palavras mais utilizadas na língua inglesa seguem a este princípio. Achou uma hipérbole para a freqüência da utilização das palavras. Foi um trabalho não muito científico e sem grande peso.

G. A. Miller se utilizou de um macaco para digitar palavras aleatoriamente, seguindo a somente duas restrições: a) A barra de espaço teria a probabilidade p e as demais letras 1/p; b) O macaco jamais digitaria duas vezes a barra de espaço na mesma linha. O macaco gerou uma saída cuja estatística ficou bem próxima a de Zipf. O modelo de Zipf não é preciso para uma distribuição de letras. Embora o trabalho de Zipf não seja muito preciso, ele foi o primeiro a utilizar a distribuição hiperbólica.

6. Técnicas de Compressão de Dados

Há várias formas de se comprimir dados, cada qual partindo de um princípio básico de compressão de dados que seu autor achava mais eficiente. Cada princípio possui suas vantagens, desvantagens e campo de aplicação de forma que quase nunca se pode afirmar que o princípio da técnica A é melhor ou pior que o princípio da técnica B. Um exemplo está no Código

Morse inventado em 1835 por Samuel Morse. O princípio utilizado por Morse para definir seu código foi que caracteres mais utilizados devem ser mais rápidos que caracteres menos utilizados. Desta forma o "e" utilizou somente um ponto (pulso mais rápido) e o "j" utilizou um ponto e três traços. Os numerais todos utilizaram cinco pulsos. A vantagem do código Morse está na eficiência da transmissão de mensagens. A desvantagem está na pouca versatilidade para lidar com expressões matemáticas. Morse utilizou um modelo de contexto finito, estático e de ordem zero. O escopo do código Morse ficou na transmissão de mensagens no telégrafo e uso militar, mas é história e não possui mais utilização.

6.1. **Código de Shannon-Fano.** O princípio do código de Shannon-Fano é parecido com o código Morse por partir do mesmo princípio, mas adaptado à tecnologia computacional. A base deste código é representar os símbolos mais prováveis com menor número de bits. A principal vantagem está na representação da mensagem com um número pequeno de bits. A desvantagem está na administração de redundâncias. O algoritmo básico deste código é:

Listar todos os símbolos em ordem decrescente de probabilidade;

- Dividir a lista em duas partes "iguais";
- Começar a codificação da primeira parte com 0 e a da segunda parte com 1;
- Continuar recursivamente até que cada subdivisão possua apenas 1 símbolo.

Pode ocorrer o fato de mensagens equiprováveis possuam tamanhos diferentes. A redundância está no fato de ocorrerem códigos possíveis que não representem mensagem alguma. A redundância é inevitável pois caso o código 011011 fosse apresentado e 01 representasse A; 104 representasse B, 11 representasse C e 011 representasse D seria impossível distinguir DD de ABC. Devemos destacar que o código de Shannon-Fano também se baseia no modelo não adaptativo (cabendo a necessidade de se ler a mensagem toda para saber a distribuição correta das probabilidades) e utiliza um número fixo de bits para representar cada símbolo.

6.2. **Código de Huffman.** O código de Huffman apresenta uma reunião de dois princípios. O primeiro princípio é o utilizado pelo código de Shannon-Fano de representar as mensagens mais prováveis com menor número de bits e também utiliza o princípio que representar as duas mensagens menos prováveis com o mesmo número de bits. Possui as mesmas vantagens e desvantagens do caso anterior, mas com algumas diferenças impostas pela segunda condição. O algoritmo básico do código de Huffman é:

- Pegar os dois símbolos menos prováveis e represente um deles como sendo final 0 e o outro como sendo final 1;
- Juntar os dois símbolos, como se fossem um só, e some suas probabilidades;
- Repita o processo até que se tenha somente um símbolo ($p = 1$), achando-se assim a representação de cada símbolo.

Supondo um exemplo no qual um texto possua quatro símbolos: A, B, C e D com respectivamente as probabilidades $0,3$; $0,3$; $0,2$ e $0,2$. Pelo algoritmo D seria representado por final 1 e C por final 0, juntos C e D teriam probabilidade $0,4$. Neste caso as piores probabilidades seriam A e B que seriam representadas por final 0 e final 1 respectivamente. Juntas teriam a probabilidade $0,6$. Neste ponto os dois símbolos menos prováveis seriam $C+D$ e $A+B$. Representando $A+B$ por final 0 e $C+D$ por final 1. Neste ponto há somente um símbolo $A+B+C+D$ com o final do processo. Neste ponto a representação de A, B, C e D seriam respectivamente 00; 01; 10 e 11. O B é representado por 01 pois o 1 seria a representação do B sozinho, quando este apareceu pela primeira vez como símbolo menos provável e o 0 seria a representação do $A+B$. O conceito de final 0 e final 1 do primeiro passo do algoritmo significa que a representação do símbolo em questão termina com o bit 0 ou o bit 1, conforme foi atribuído pelo algoritmo. Notar a diferença de representação se as probabilidades forem respectivamente para A, B, C e D: $0,5$; $0,2$; $0,2$ e $0,1$ as representações poderiam ser respectivamente 0; 11; 100 e 101 ou 1; 00; 010 e 011, dependendo da forma de atribuir 0 ou 1 dada pelo algoritmo.

Na média pode-se afirmar que o código de Huffman é mais eficiente (menor redundância) que o código de Shannon-Fano, mas há casos em que ocorre o contrário.

O algoritmo de Huffman pode ser utilizado de forma adaptativa, tendo por solução trivial recriar a árvore após a transmissão de cada símbolo porém isto consumiria muito tempo e recurso. Uma solução prática foi criar a seguinte estrutura: Dado que no código de Huffman a árvore define os códigos somente nas folhas uma alternativa está em colocar na árvore de Huffman uma condição a mais: a quantidade de vezes que o símbolo aparece (peso) e colocar nos nós internos (não folhas) a soma dos pesos nos nós descendentes. Haveria neste caso uma combinação entre o transmissor e o receptor para os símbolos ainda não transmitidos e uma folha especial, com peso 0 para representar os símbolos ainda não transmitidos. Quando um símbolo for ser transmitido pela primeira vez é transmitido o código de símbolo não transmitido seguido pelo código pré combinado entre o transmissor e o receptor para este símbolo do alfabeto. Então este símbolo é retirado do conjunto de símbolos ainda não transmitidos e é colocado no seu nó o peso 1. A partir daí cada vez que este símbolo for transmitido seu peso é acrescido de 1. Toda vez que se transmite um símbolo é feito o ajuste de pesos nos nós internos da árvore e, se for o caso, feito um ajuste na árvore. Desta forma continuam-se os custos de manutenção dos pesos nos nós internos da árvore, mas a mesma somente será recalculada se for o caso. Como no caso das linguagens naturais há uma distribuição estatística que normalmente funciona a tendência é de a partir de certo ponto a árvore estabilizar e o número de nós internos a serem alterados diminuir pois os símbolos mais freqüentes ficam no topo da árvore.

Os dois códigos anteriores são baseados em princípios da representação de cada símbolo por um número fixo de bits e também no suposição que é eficiente representar os símbolos/mensagens mais freqüentes com um número menor de bits é eficiente. Mas os algoritmos de compressão devem ser adaptados para a realidade da mensagem que pretendem comprimir. Vejamos o exemplo:

Suponha que a mensagem a ser representada possui 90.000.000 de números inteiros menores que 1.000.000.000. Neste caso o número de palavras é muito grande. Mas supondo ainda que estes números têm a propriedade que em 98% dos casos a diferença de um número para seu sucessor seja menor que 20, podendo variar para mais ou para menos. Dadas estas condições de contorno o algoritmo abaixo atenderia bem à representação da mensagem.

- O primeiro número é codificado ou decodificado em código binário em seu tamanho máximo possível (30 bits).
- A partir do segundo número a representação assume a seguinte regra para o codificador:
  - Pega-se o próximo número e subtrai do número atual.
  - Se a diferença for maior ou igual a zero e menor que 32 representar este número utilizando 6 bits;
  - Se a diferença for menor que zero e maior que -31 ligar o próximo bit e representar o número nos 5 bits seguintes;
  - Se não ocorrer nenhum dos dois casos anteriores ligar os seis próximos bits e representar o número em binário utilizando os 30 bits seguintes.
- A partir do segundo número a representação assume a seguinte regra para o decodificador:
  - Pegar os seis próximos bits e verificar:
  - Se o primeiro estiver desligado os próximos 5 bits representam o número em base 2 que deverá ser somado ao número atual para se saber qual o próximo número;
  - Se os próximos 6 bits estiverem todos ligados a representação do próximo número estará em código binário nos próximos 30 bits;
  - Se não ocorrer nenhum dos dois casos anteriores (o primeiro bit está ligado e pelo menos um dos outros 5 bits estiver desligado) neste caso os próximos 5 bits seguintes ao bit inicial ligado indicam o número binário que deverá ser subtraído do número atual para se saber qual o próximo número.

Este algoritmo criado sob a encomenda do escopo acima pode representar um dado número se utilizando de 30 (primeiro número), 36 ou 6 bits, conforme a situação na qual ele esteja. Mas como em 98% dos casos a diferença é menor que 20 em 98% dos a representação do número utilizará apenas 6 bits para representar cada mensagem. Uma das vantagens deste algoritmo está no fato de permitir a criação de um índice simples (a cada mil números por exemplo) e que permitiria um acesso relativamente direto ao número 7.235.415 da mensagem que está sendo comprimida. Uma das desvantagens da compressão está no fato de quase nunca ser possível o acesso direto ao dado. Devido ao tipo de mensagem e ao algoritmo utilizado é possível a criação de um índice para o acesso quase direto. Mas no quotidiano as mensagens quase nunca são tão bem comportadas para que este algoritmo seja recomendável para uso geral. Caso exista alguma regra na formação da seqüência do DNA, a determinação desta regra tornaria a cadeia bem mais previsível e diminuiria sua entropia. Porém se há tal regra ainda não foi descoberta.

6.3. **Compressão Aritmética.** A Compressão Aritmética é um exemplo que mostra a diversidade de modelos possíveis. O princípio no qual se baseia este algoritmo é completamente diferente dos utilizados nos algoritmos anteriores, embora possua a mesma finalidade. O princípio básico envolvido no algoritmo de Compressão Aritmética é que cada mensagem pode ser representada por um número no intervalo entre [0, 1). A conseqüência deste princípio é: Quanto maior a mensagem menor o intervalo necessário para representá-la e maior a quantidade de algarismos para representar este intervalo.

Uma das principais diferenças quanto aos algoritmos anteriores está na necessidade de haver um símbolo para representar o final da mensagem pois é a única forma de se poder decodificar corretamente. O algoritmo possui os seguintes passos:

- É dada uma distribuição de probabilidades inicial aonde são colocados os símbolos, incluindo-se aí o símbolo de final de mensagem;
- Uma vez codificado algum símbolo é feito o ajuste do intervalo para a inclusão do novo símbolo;
- Repete-se o processo até que se chegue ao símbolo de final de mensagem.
- Escolhe-se convenientemente um valor dentro do intervalo final.

O decodificador trabalha da mesma forma, recebe a distribuição de probabilidades idêntica à utilizada pelo codificador e de acordo com a probabilidade da mensagem recebida vai identificando cada símbolo enviado, até chegar ao símbolo de final de transmissão. Caso fosse implementada uma probabilidade decimal a mensagem *ELISA FONSECA HORTA* seria representada como $0,33836329579528298755446$ gastando 23 algarismos significativos e exigindo uma precisão da máquina de alguns algarismos a mais para evitar perda na precisão do processo. Feita a devida conversão de inteiros para a base 2 necessitaríamos de uma precisão em torno de 100 bits para conseguir trabalhar uma mensagem de 20 caracteres. A distribuição estatística para obter este valor está no apêndice 1. Destacamos que se tratar uma mensagem curta, de apenas 20 caracteres e 14 símbolos.

Desta forma seria inviável se implementar a compressão aritmética. Soluções independentes foram achadas quase simultaneamente por Rubin, Guazzo e Rissanen, no final da década de 70. A visão moderna da compressão aritmética utiliza números inteiros e leva em conta transmissão e recepção incremental; problemas como *overflow* e *underflow* e eficiência do algoritmo.

O conceito de trabalhar com números inteiros e transmissão/recepção incremental também está detalhada no apêndice 1. Mas o conceito básico é o seguinte: As probabilidades podem ser tratadas através da utilização de intervalos de números inteiros. Por outro lado a transmissão e recepção incremental se baseia no seguinte argumento: Dado que o bit de mais alta ordem dos números que representam o menor valor e o maior valor são os mesmos, estes não interferem mais no processo, podendo ser transmitidos.

O problema de trabalhar com aritmética de ponto fixo está que quanto maior a mensagem maior deverá ser a precisão do número a ser trabalhado para codificar/decodificar a mensagem. Devem ser tomadas medidas para evitar o *overflow*, quando se trabalha com um número maior do que o que se consegue representar, e o *underflow*, quando o intervalo entre valor mínimo e valor máximo não é grande o suficiente para representar um determinado código. No processo da compressão aritmética o codificador e o decodificador trabalham da mesma forma, utilizando-se das mesmas estatísticas e probabilidades. A mesma consistência para evitar *underflow/overflow* aplicada no codificador deve ser aplicada no decodificador para que seja possível decodificar corretamente. Uma forma de evitar o *underflow* é garantir que o tamanho do intervalo seja sempre maior que o tamanho do símbolo de maior freqüência. Toda vez que o tamanho do intervalo ficar menor que o tamanho para representar o símbolo de maior freqüência soma-se um valor fixo ao máximo e subtrai-se um valor fixo ao mínimo, garantindo-se com isto a condição suficiente para não haver *underflow*. O decodificador fará o mesmo teste e mesmo ajuste. É fato que trabalhar com binários não é intuitivo para a maioria das pessoas e há regras específicas para números binários representados em um computador que não são válidas para números decimais fora deste universo. Um exemplo de como se evitar *underflow* para o universo dos números representados na base 10 com precisão de 12 algarismos:

Supondo que para representar o símbolo de maior freqüência sejam necessários 4 algarismos. A regra de testar o intervalo poderia ser dita da seguinte forma: seja T_i o tamanho do intervalo, L_i o limite inferior e L_s o limite superior, se $T_i < 10.000$ e $L_i > 10.000$ então $L_i \leftarrow L_i - 10.000$ senão se $T_i < 10.000$ e $L_i \leq 10.000$ então $Ls \leftarrow Ls + 10.000$. Como o teste seria feito sempre nas mesmas condições isto garantiria que não ocorreria a situação de *underflow*.

Uma observação é que toda vez que o limite inferior e superior estiverem na mesma metade de um número binário haverá a existência de maior bit significativo igual e a conseguinte transmissão deste bit e os ajustes devidos para o maior intervalo e menor intervalo, o que obviamente não vale para números decimais(4×10^{11} e 3×10^{11} são menores que 5×10^{11} estariam na mesma metade de um número na base 10 de 12 algarismos mas não definiriam um número a ser transmitido).

A mesma regra se aplica ao *overflow*. A implementação em binários garante que não haverá *overflow* por se utilizar da freqüência máxima permitida e outros conceitos que não cabem detalhar. Há limitações na implementação do código aritmético que garantem as condições necessárias para evitar *underflow* e *overflow*, mas não serão detalhadas.

Ao final da mensagem é necessário transmitir o símbolo de fim de mensagem seguido de tantos bits quantos forem necessários para garantir que a cadeia está no range de final de mensagem.

A Compressão aritmética pode ser implementado de forma adaptativa. Neste caso é combinada uma distribuição de probabilidades inicial para cada símbolo e cada símbolo ganha também um contador de ocorrências. Cada vez que o símbolo ocorre o contador é incrementado em 1 e então é recalculada as distribuições probabilísticas. Para facilitar o cálculo das probabilidades pode-se utilizar contadores acumulativos. Manter os símbolos em ordem de freqüência otimiza o processo pois diminui a quantidade de atualizações dos índices acumulativos. Há implementações específicas para a compressão aritmética para grandes alfabetos e também para códigos binários.

6.4. **Blending.** Blending é uma técnica de compressão que permite que diferentes ordens sejam utilizadas simultaneamente para determinar a probabilidade do próximo símbolo. Nesta técnica, essencialmente adaptativa, cada ordem ganha um peso w e a ordem -1 é utilizada para permitir a utilização dos símbolos do alfabeto ainda não utilizados. A soma dos pesos de cada ordem utilizada deve somar 1. A técnica de blending normalmente é utilizada para fornecer as estatísticas para algum algoritmo como o algoritmo de compressão aritmética. Duas características importantes do blending são a existência de um caractere de escape e um limite máximo para a ordem a ser adotada. Sendo utilizada normalmente a letra m para indicar o limite máximo para a ordem a ser adotada. Há várias formas de se calcular a probabilidade do escape e a probabilidade do escape influencia na definição dos pesos de cada ordem. No exemplo adotado por Bell, Cleary e Witten [1] o cálculo dos pesos é feito em função da probabilidade do escape da ordem superior e do escape da própria ordem e está detalhado no apêndice 2. A grande dificuldade da estrutura do blendig pura está na forma complexa para se manter as probabilidades e os pesos de forma dinâmica. Uma solução adotada está na adoção da simplificação, chamada exclusão, que considera a probabilidade de cada símbolo ocorrer somente para a ordem mais alta a qual ele seja previsto. Vários algoritmos foram criados utilizando blending, mas será dado destaque somente ao ppm sigla em inglês para previsões com batida parcial (prediction with partial match). Há três algoritmos básicos que utilizam o PPM. A principal diferença entre eles está na forma de se calcular o caractere de escape.

6.4.1. *PPMA.* Sayood [8] utiliza o PPMA em conjunto com a compressão aritmética para exemplificar as tabelas de controle necessárias para o algoritmo. No PPMA a técnica utilizada para calcular os escapes consiste em para cada ordem, exceto a ordem -1, incluir um caractere de escape que é transmitido (utilizando a compressão aritmética) quando naquela ordem o próximo símbolo ainda não ocorreu. Este modelo é muito prático quando há pouca necessidade de se recorrer ao caractere de escape pois deixa relativamente pouca probabilidade para a utilização do caractere de escape.

6.4.2. *PPMB.* O segundo algoritmo o PPMB utiliza para calcular o escape o fato de a primeira vez que um símbolo ocorre ele não é incluído nas probabilidades esta ocorrência é adicionada ao contador de escapes. Desta forma o contador de escapes terá exatamente o número de símbolos que ocorreu e cada símbolo terá seu contador reduzido na unidade em cada ordem. A vantagem deste método está que casos anômalos, que somente ocorrem uma vez, não terão estatística, deixando a distribuição de probabilidades direcionada aos eventos que ocorreram mais de uma vez.

6.4.3. *PPMC.* O PPMC assim como o ppmb utiliza para o contador de escape o número de símbolos que apareceram para a ordem mas sem subtrair em um o contador de cada símbolo. Notar que qualquer que seja o algoritmo utilizado a distribuição de probabilidades sofrerá a exclusão para que cada símbolo tenha somente a probabilidade da mais alta ordem na qual ele aparecer. Este é o princípio do blendig que serve para indicar as probabilidades para algum algoritmo como por exemplo a compressão aritmética e nem tanto para por si efetivar a compressão dos dados.

6.5. **Dicionários.** As técnicas baseadas em dicionário. A compressão de dados baseada em dicionários parte do princípio que grupos de símbolos consecutivos, as frases, estão vinculados a índices de algum dicionário e o índice, ocupa menos espaço que a frase a qual está vinculado. O dicionário é uma lista de frases que se espera sejam muito utilizadas. Os dicionários possuem uma implementação eficiente pois os índices podem ser criados no tamanho de palavras do computador evitando-se assim a manipulação de bits e os alinhamentos. Formalmente um dicionário $D = (M, C)$ é um conjunto finito de frases M e uma função C que faz um mapeamento de M em um conjunto de códigos.

Assim como na compressão estatística a compressão por dicionários pode ser estática, semi adaptativa ou adaptativa. Defini-se por *parsing* a tarefa de dividir o texto em frases. Em português o *parsing* pode ser chamado de fraseamento. Uma das estratégias mais comuns para se fazer o fraseamento está no fraseamento guloso que consiste em encontrar no texto a maior frase que pertence a M e substituí-la por seu índice. Lamentavelmente esta estratégia não é necessariamente ótima. A tarefa do fraseamento pode transformada no problema de encontrar o caminho mais curto em um grafo. Embora não seja ótimo o fraseamento guloso é o mais utilizado.

Codificadores estáticos baseados em dicionários: O mais comum é o codificador de digramas e funciona da seguinte forma: Olha-se para os dois próximos algarismos a serem codificados, se o digrama fizer parte do dicionário este será substituído por sua representação, caso contrário grava-se o primeiro caractere do digrama e se repete a operação até o final do texto. Este processo funciona bem porque na língua inglesa há somente 96 caracteres de texto e os (256 - 96) 160 possíveis combinações de um byte podem ser utilizados para representar o 160 digramas mais freqüentes do texto. Já na língua portuguesa há menos eficiência neste processo, pois além das letras do alfabeto inglês (o K, W e Y são usados em palavras como windsurf, byte e abreviações como kwh) há o Ç, e as letras acentuadas com trema (¨); acento grave (‘); acento agudo (´); acento circunflexo (ˆ); til (˜), que podem estar tanto em maiúscula como em minúscula. Eventualmente pode-se querer colocar no dicionário trigramas, tetragramas ou até pentagramas mais freqüentes como ‘ th’ na língua inglesa, mas há maior perda de tempo no processo de compressão de dados. Este processo também pode ser ampliado para trabalhar com dicionário estáticos de n-gramas criando-se palavras com 4, 8 ou 12 bits de tamanho e os primeiros 4 bits indicarão o tamanho da entrada do dicionário.

Codificadores semi-adaptativos baseados em dicionários: Para textos técnicos há expressões que ocorrem com bastante freqüência. Neste texto as palavras “compressão”, “freqüência” e “adaptativos” ocorrem com bastante freqüência. Há também frases como “baseados em dicionários” que podem ser candidatas a aparecer no dicionário. A grande vantagem neste caso está que as frases normalmente são bem maiores que os índices utilizados para representá-las e com isto se pressupõe uma boa compressão de dados. O problema está em determinar um dicionário ótimo para um dado texto. É um problema NP-Completo no tamanho do texto o que dificulta bastante o trabalho. O que tem sido feito na prática: Preenche-se inicialmente o dicionário com os digramas mais freqüentes.

Quando o trigrama mais freqüente for mais freqüente que algum digrama ele é inserido e neste ponto se analisa se os digramas que o formam forem pouco freqüentes fora do trigrama estes são excluídos do dicionário. Exemplo: Suponha que os digramas " a" e "a " sejam muito freqüentes. Ao se inserir o trigrama " a " verifica-se as ocorrências dos digramas anteriores fora do trigrama " a ". Se forem pouco freqüentes estes serão removidos (ou se somente " a" for pouco freqüente fora de " a " então somente este será removido do dicionário).

Codificadores adaptativos baseados em dicionários: Após 1977 e o aparecimento do codificador ZIV-LEMPEL falar em modelos adaptativos baseados em dicionários se reduz basicamente a falar do codificador ZIV-LEMPEL e das diversas variações deste modelo. Estes codificadores são chamados LZ qualquer coisa sendo que os criados pelos autores Jacob Ziv e Abraham Lempel foram os LZ77 e LZ78 por terem sido formulados nos anos de 1977 e 1978. No livro Text Compression [1] há referências a 12 métodos LZ dos quais alguns. Os códigos Ziv Lempel possuem a seguinte característica: Utilizam como dicionário a própria entrada de texto anteriormente codificada. É claro que para o início do processo o dicionário está vazio e o primeiro caracter codificado é de fato o primeiro caracter. A partir deste ponto começa a ocorrer o seguinte fato: Para cada seqüência "s" de caracteres candidatos à serem codificados é olhado no texto original já codificado se qual a maior seqüência de caracteres comuns a partir do início de "s" e se for o caso se faz a referência a esta seqüência. Suponha um pequeno texto de poucos caracteres: "Aaaaaaaaaaaaaaaaaaaaaaaaaaabababababababababababab" (A - a ocorrendo 26 vezes - ba ocorrendo 11 vezes - b ocorrendo 1 vez). Neste texto a compressão iniciaria com A, na seqüência restante não há ocorrências anteriores de forma que a codificação colocaria um a tendo a seqüência Aa. A partir deste instante começa haver algo interessante, pois para os próximos 25 caracteres há uma repetição da seqüência definida a partir do segundo caractere. Neste caso a codificação ficaria $Aa(2,25)$ indicando que os próximos 25 caracteres (tamanho da maior seqüência de caracteres comuns a partir do início do texto a ser codificado) repetem a seqüência iniciada na posição 2 do texto original. Nota-se que pode haver sobreposição entre a seqüência a codificar e o texto já codificado. O próximo caracter a ser codificado é um b que ainda não ocorreu anteriormente. Neste caso codifica-se o b, ficando-se com a seqüência $Aa(2,25)b$. Neste ponto foram codificados 28 caracteres mas ainda faltam codificar os restantes 22 caracteres. Mas estes 22 caracteres repetem a seqüência iniciada na posição 27, ficando portanto a representação dos 50 caracteres originais do texto em $Aa(2,25)b(27,22)$. Este texto foi escolhido sob medida para exemplificar o mecanismo da codificação Ziv Lempel. Embora a ocorrência de taxas de compressão desta eficiência sejam raras o algoritmo é eficiente e funciona muito bem para a língua inglesa. A grande vantagem deste método é que não se faz necessário representar o dicionário pois este está no próprio texto codificado anteriormente seguido do texto a ser codificado. As variações dos códigos LZ estão no tamanho do texto anterior a ser olhado, no texto a frente a ser codificado e na estrutura para a busca da maior seqüência de caracteres comuns ao dicionário e ao início do texto a ser codificado. Há diversos algoritmos baseados nos códigos LZ.

6.5.1. *LZ77.* Este algoritmo consiste em olhar a janela anterior (inicializada normalmente como espaços em branco) e verificar qual o maior tamanho de cadeia de caracteres já ocorreu na janela e representar o deslocamento (quantos caracteres anteriores ao primeiro a codificar) seguido pelo tamanho da maior cadeia que já ocorreu e seguida do próximo caracter que não coincidiu com a cadeia. Desloca-se a janela do tamanho da cadeia (que pode ser 0) + 1. A janela neste caso é composta de duas partes: a janela anterior que é composta exclusivamente de caracteres já codificados e a seqüência de caracteres a codificar. Quando se está consultando a janela pode haver sobreposição entre o que já foi codificado com o que falta codificar para se definir a maior cadeia que já ocorreu, mas necessariamente deve-se haver um deslocamento positivo. Caso não haja esta cadeia, costuma-se representar o próximo caracter como (0, 0, "x") sendo "x" o próximo caracter. Neste caso os dois primeiros zeros indicam que na janela anterior não há o primeiro caracter da seqüência de caracteres a representar e o segundo 0 indica que o tamanho da cadeia que coincidiu com a seqüência de caracteres a representar. As implementações variam no tamanho da janela, sendo usual o tamanho da janela de 8192 caracteres sendo entre 10 e 20 o número de caracteres a serem codificados e os demais destinados à janela anterior. Um exemplo simples: codificar AATAAATAATATAAA olhando-se 7 caracteres para trás e 5 a frente (tamanho de janela de 8 caracteres): $(0, 0, A)(1, 1, T)(3, 2, A)(4, 3, T)(5, 4, A)$. Neste caso a seqüência final teria um tamanho de 10 bytes uma vez que tanto o deslocamento quanto o tamanho da cadeia podem ser representados em 4 bits. O LZ77 é muito utilizado, são baseados neste algoritmo PKZip, Zip, LHarc, PGN, gzip e ARJ.

6.5.2. *LZ78.* Este algoritmo consiste em criar um dicionário em tempo de execução. Cada entrada no dicionário consiste em uma entrada anterior seguida de um caractere. Considere a notação d_i como sendo a i-ésima entrada no dicionário. O índice 0 denota a entrada vazia. Assim a codificação de $AATAAATAATATAAA$ seria:
$d_1 \leftarrow (0, A), d_2 \leftarrow (1, T), d_3 \leftarrow (1, A), d_4 \leftarrow (2, A), d_5 \leftarrow (4, T), d_6 \leftarrow (3, A)$. A primeira entrada do dicionários seria A, a segunda AT, a terceira AA, a quarta ATA, a quinta $ATAT$ e a sexta AAA. Na prática não é possível criar um dicionário de tamanho indefinido. Chega-se em um certo ponto em que temos que jogar fora o dicionário e começar novamente como se não houvesse nada anterior. Um ponto de pega está quando uma frase se torna repetitiva no texto. Suponha que a frase "Alternativa: " ocorra 100 vezes em um texto: Isto provocaria 100 entradas no dicionário muito parecidas, causando um custo relativamente caro para se tratar a mesma entrada. Por outro lado há um ganho efetivo nas buscas utilizando-se uma estrutura *trie*

que consiste de uma árvore com vários caminhos na qual cada nó tem por pai o nó com uma letra a menos. No exemplo acima a raiz seria um nó vazio que teria um só filho o nó A (nó 1). O nó 1 teria por filhos os nós 2 (AT) e 3 (AA). Já o nó 2 teria por filho o nó 4 (ATA) e o nó 3 teria por filho o nó 6 (AAA). O nó 4 teria por filho o nó 5 ($ATAT$) e os nós 5 e 6 não teriam filhos.

6.5.3. *LZW*. É uma variação importante do LZ78. Trata-se de inicializar o dicionário com os símbolos do alfabeto e a partir daí ir incrementando o dicionário com as novas frases, até que o mesmo esteja cheio, quando então passa a não mais receber frases. A vantagem deste método está na não necessidade de transmitir um caractere a mais após cada apontador. No exemplo cuja mensagem $M = aababababaaa$. O dicionário é inicializado com $d_0 \leftarrow a$ e $d_1 \leftarrow b$. A saída deste caso é: 001352. Como foram obtidos estes valores? Os dois primeiros zeros correspondem aos símbolos pré inicializados a. Mas ao se ler o segundo a cria-se uma nova entrada no dicionário $d_2 \leftarrow aa$. O 1 é obtido do símbolo pré inicializado b e o dicionário recebe $d_3 \leftarrow ab$. O próximo símbolo 3 é obtido do último símbolo inserido no dicionário $d_3 \leftarrow ab$ e o dicionário recebe $d_4 \leftarrow ba$. O próximo símbolo d_5 tem uma característica interessante, ele ainda não está definido no dicionário mas aparece na codificação. Sabe-se que do início da codificação os dois próximos caracteres são ab. Por convenção quando uma entrada no dicionário está sendo formada o último símbolo é igual ao primeiro e neste caso $d_5 \leftarrow aba$. O 2 final sai diretamente do dicionário após esta codificação o dicionário recebe $d_6 \leftarrow abaa$, mas a codificação já chegou ao final. Como todos os outros métodos este também tem suas vantagens e desvantagens, cabendo a quem vai efetivar a compressão escolher o método mais adequado às suas necessidades.

7. O Estudo da Entropia do DNA em Artigos Científicos

Os artigos que tratam da Entropia do DNA seguem duas linhas distintas. A primeira utiliza as bases e a segunda os aminoácidos. O primeiro artigo que será visto trabalha na primeira linha, já o segundo artigo na segunda linha. Loewenstern e Yianilos [6] fizeram um trabalho específico para calcular a entropia de seqüências de DNA, o CDNA. Este trabalho se baseia nos seguintes aspectos: Pega-se uma cadeia de DNA de tamanho suficiente grande. Divide-a em 8 segmentos de mesmo tamanho. O primeiro segmento é dividido em seqüências de tamanho fixo w (por exemplo 20 nucleotídeos). Para cada seqüência do primeiro segmento é feito um cálculo de h: distância de Hamming (número de bases que não coincidiu com a seqüência original) para cada seqüência de tamanho w em cada um dos 7 segmentos restantes. É então feita a estatística de ocorrência da próxima base para cada h. A partir daí calcula-se a entropia do segmento original. Repete-se o processo para os demais segmentos. A entropia da seqüência de DNA é a média das entropias obtidas. Cabe destacar que para pequenos valores de h o número de ocorrências superou em muito a expectativa teórica para uma distribuição aleatória. Já para grandes valores de h o número teórico de ocorrências foi maior que o encontrado, o que mostra que as seqüências de DNA não são bases ao acaso e há sim uma organização bem maior que o aleatório. Outro fato relevante é que para os menores valores de h ($0 \leq "h" \leq 5$) a porcentagem de acerto da próxima base é bem maior que para os maiores valores de h ($16 \leq "h" \leq 20$). É bom lembrar que a probabilidade de acerto da próxima base em uma distribuição aleatória é de 25%. Isto mais uma vez induz à uma organização bem melhor que a aleatória para uma cadeia de DNA. Segue abaixo uma cópia da tabela 1 retirada deste trabalho [6].

TABLE1. PREDICTED VERSUS OBSERVED MATCHES (HUMRETBLAS. WINDOW SIZE 20)

Hamming distance	Expected nuber of matches	Observed number of matches	Percentage hit at least once	Percentage correct givem 1 + hits
0	1.44×10^{-7}	0.387	4.4	90.91
1	8.61×10^{-6}	0.143	7.2	83.77
2	2.45×10^{-4}	0.278	11.0	78.19
3	4.42×10^{-3}	0.505	17.6	66.00
4	0.0563	1.184	33.2	45.91
5	0.541	4.148	78.2	35.86
6	4.01	18.759	98.3	31.90
7	24.3	80.69	100	29.19
8	119	304.9	100	28.35
9	475	986.0	100	28.09
10	1,566	2,713	100	27.86
11	4,271	6,286	100	27.68
12	9,609	12,300	100	27.53
13	17,740	20,172	100	27.33
14	26,610	27,387	100	27.09
15	31,932	30,356	100	26.86
16	29,936	26,813	100	26.57
17	21,131	18,200	100	26.28
18	10,566	8,931	100	25.94
19	3,337	2,826	100	25.69
20	500	434.5	100	25.43

Contexts of length 20 from first 1/8 were matched to contexts in subsequent 7/8 of the gene.

Segue abaixo os dados da entropia para a seqüência de DNA HUMRETBLAS também retirada deste trabalho [6] (tabela 2 linha 9) que trata da estimativa de entropia (Bits / Nucleotídeos). H1, H4 e H6 se referem a estimativas de entropias convencionais.

Seqüência	Tamanho	Unix Compress	H1	H4	H6	CDNA	CDNA Compress
Humretblas	180.388	2,14	1,95	1,90	1,90	1,66	1,75

O que foi mostrado por este artigo é que a Entropia do DNA está bem abaixo do que se tem sido conseguido com os processos convencionais de compressão universais. O fato de se permitir diferenças entre as seqüências de bases de DNA (distância de Hamming) não ajudou muito na previsibilidade, embora tenha colocado a entropia bem abaixo do que se tem conseguido com os algoritmos utilizados para a linguagem natural. A conclusão que se chega é que no "universo" das seqüências de bases de DNA palavras e frases aparecem pouco ou até nem aparecem, o que dificulta bastante a estimativa de sua entropia. Outro aspecto importante deste trabalho está na comparação de previsões entre seqüências distintas de DNA. A tabela 5 deste trabalho mostra que ao se treinar a entropia no grupo "mammalian"e se utilizar os parâmetros de previsão no grupo "yeast"e vice-versa a entropia ficou pior (maior) que o previsto para o cada grupo. A conclusão que se pode chegar é que as características das cadeias de DNA divergem de DNA para DNA e que palavras e frases não fazem parte destas características. Isto nos induz a pensar que ao contrário das bases e aminoácidos que são universais as cadeias de DNA possuem suas estruturas próprias e peculiares.

Outra consideração deste trabalho é que o CDNA se utiliza apenas de um domínio do conhecimento prévio do DNA: As seqüências de DNA naturais possuem mais repetições próximas do que se pode esperar de uma seqüência aleatória. Há outros conhecimentos prévios a respeito do DNA que podem ser utilizados para se fazer uma estimativa mais acurada da entropia de seqüências de DNA.

Conforme foi desenvolvido no artigo de Loewenstern e Yianilos [6], a estimativa da Entropia do DNA pode sem bem melhorada. O mesmo já não ocorre com relação às proteínas. O próximo artigo que a ser analisado é o artigo de Nevill-Manning, e Witten [7].

Este artigo leva em conta que as mutações genéticas são aleatórias o que levaria a uma menor compressibilidade das regiões exon do DNA. Embora haja uma questão filosófica por trás deste fato cabe destacar que neste não se conseguiu comprimir de forma significativa as proteínas o que pode significar que os códons das proteínas são imprevisíveis. São conhecidas da biologia as probabilidades de mutações de um códon em um dado período. O fator de probabilidades de mutações genéticas dos aminoácidos em função do tempo (62 milhões de anos) estão na tabela abaixo extraída do artigo de Nevill-Manning, e Witten [7]

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	17	4	3	3	3	3	5	9	2	5	7	5	2	3	3	10	6	1	2	8
R	5	21	5	4	1	6	6	4	3	3	6	15	2	2	2	5	4	1	2	4
N	5	5	19	10	1	4	6	8	4	3	4	6	1	2	2	8	6	0	2	3
D	5	4	9	25	1	4	11	6	2	3	4	6	1	2	3	7	4	0	1	3
C	8	2	2	2	32	2	2	4	1	6	8	3	2	3	2	6	5	1	2	7
Q	6	8	5	5	1	12	12	4	3	3	5	10	2	2	3	6	5	1	2	4
E	6	6	5	11	1	8	17	4	3	3	4	9	1	2	3	6	4	1	2	4
G	11	3	5	5	1	2	4	34	2	3	4	5	1	2	2	7	4	1	2	3
H	5	6	7	4	1	5	6	4	22	3	5	6	2	4	2	5	3	1	7	3
I	5	2	2	2	2	2	2	2	1	16	19	3	4	5	2	3	5	1	2	20
L	6	3	2	2	2	2	2	3	1	14	23	3	6	7	2	3	4	1	3	12
K	7	12	5	5	1	6	8	5	2	3	5	16	2	2	3	6	5	1	2	4
M	6	3	2	2	2	3	3	3	2	11	21	4	9	5	2	4	4	1	2	10
F	4	2	2	2	1	1	2	3	2	8	14	2	3	24	1	3	3	2	11	7
P	7	3	3	4	1	3	5	5	2	3	5	5	1	2	33	6	5	0	2	4
S	12	4	6	6	2	4	6	8	2	3	5	6	2	2	3	12	9	1	2	5
T	8	4	5	4	2	3	5	5	2	6	7	5	2	3	3	11	14	1	2	8
W	4	3	2	2	1	2	3	4	2	4	7	3	2	9	1	3	3	33	9	4
V	5	3	3	2	1	2	3	3	6	5	8	4	2	16	2	4	3	3	19	6
Y	8	3	2	2	2	2	3	3	1	19	15	3	4	4	2	4	6	1	2	16

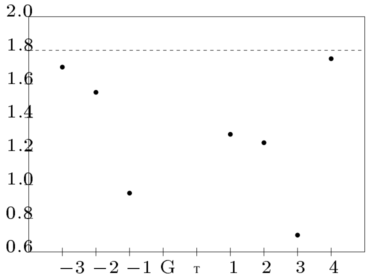
Este trabalho utiliza do processo evolutivo das mutações dos aminoácidos para a partir daí, utilizando um processo mais sofisticado que o "blending" chamado de "CP compression" tentar fazer previsões sobre a seqüências de aminoácidos da proteína. O processo é mais sofisticado que o blending pois permite que a variação ocorra para cada possível mutação na seqüência de códons, ao passo que no blending o processo somente pode ser diferente no extremo da seqüência (se não for igual volta para a ordem imediatamente anterior). Os resultados obtidos pelo "CP compression" foram um pouco melhores que os algoritmos de compressão genéricos, mas na opinião dos autores não se trata de um sucesso, mas tende a comprovar que os exons são menos compressíveis que os introns.

O terceiro artigo que referendado é o artigo de Lanctot, Li e Yang [5]. Neste artigo é feita uma rápida análise sobre os algoritmos para estimativa de Entropia do DNA, dentre eles o CDNA de Loewenstern e Yianilos [6] e em seguida é apresentada a base de seu algoritmo GTAC (Grammar Transform Analysis and Compression). Trata-se de um algoritmo baseado no modelo gramático que se mostrou eficiente na compressão de cadeias do DNA. Segue abaixo a tabela de cálculo da entropia obtido pelo GTAC. Na maioria dos casos ele foi o melhor algoritmo mas em alguns casos ele ficou aquém do CDNA.

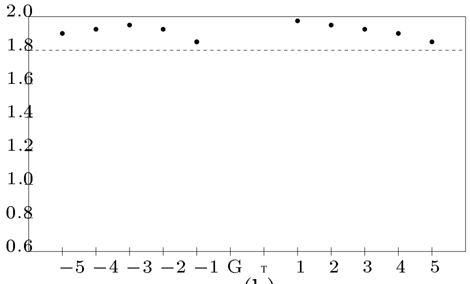
Sequence name	Sequence length	Unix compress	Bio-compress-2	CDNA compress	GTAC
PANMTPACGA	100314	2.12	1.88	1.85	1.74
MPOMTCG	186609	2.20	1.94	1.87	1.78
CHNTXX	155844	2.19	1.62	1.65	1.53
CHMPXX	121124	2.09	1.68	-	1.58
SCCHRIII	315339	2.18	1.92	1.94	1.82
HUMGHCSA	66495	2.19	1.31	0.95	1.10
HUMHBB	73308	2.20	1.88	1.77	1.73
HUMHDABCD	58864	2.21	1.88	1.67	1.70
HUMDYSTROP	38770	2.23	1.93	1.93	1.81
HUMHPRTB	56737	2.20	1.91	1.72	1.72
VACCG	191737	2.14	1.76	1.81	1.67
HEHCMVCG	229354	2.20	1.85	-	1.74

A conclusão deste trabalho foi que a utilização de códigos gramáticos torna os algoritmos universais e rodam em tempo linear. Chegou também a conclusão que o algoritmo GTAC é bom mas pode ser melhorado com a inclusão de considerações como a da batida não exata utilizada pelo CDNA.

O último artigo que será visto é o primeiro dos quatro artigos em ordem cronológica, o que mostra que o estudo da Entropia do DNA não é um modismo, mas trata de um assunto complexo que a comunidade científica vem se dedicando há anos. Este artigo de Farach; Noordewier; Savari; Shepp; Wyner e Ziv [2] a estimativa da entropia necessita ser relativamente rápida pois a região dos exons é normalmente bem menor que a região dos introns implica uma perda de convergência da estimativa da entropia a menos que o algoritmo que mede a entropia seja rápido: convirja rapidamente. Outro aspecto abordado pelo artigo está na detecção das regiões dos introns. Nos segmentos analisados pelo artigo havia 39439 cadeias iniciando com GT



(a)



(b)

mas apenas 579 regiões de introns (que também iniciam com GT). A figura 3 retirada deste trabalho mostra o cálculo da entropia das cadeias em torno do par GT, tanto para as regiões de início dos introns quanto para todas as cadeias em torno do GT.

Desta forma o algoritmo feito para cálculo da entropia se mostrou eficiente para identificar o início dos introns. Porém o mesmo algoritmo não foi eficiente para achar o final dos introns. A conclusão deste trabalho foi que houve falha na tentativa de medir a entropia característica do DNA porque foi utilizada estimativas erradas de estudos anteriores provavelmente baixa taxa de convergência dos programas que fizeram as estimativas anteriores da Entropia do DNA. O resultado principal foi que a entropia dos exons é maior que a dos introns o que de certa forma surpreendeu os autores que supunham que a aleatoriedade nos introns seria maior que nos exons (pois as mutações nos exons somente sobreviveriam se mantivessem as funções vitais dos seres vivos ativada).

Com este artigo foi possível visualizar um pouco da dificuldade de medir a Entropia do DNA e a importância do significado biológico neste estudo.

8. Conclusão

O principal problema para a correta estimativa da Entropia do DNA está na falta de conhecimento específico sobre os mecanismos que envolvem o DNA. Basta dizer que a finalidade dos introns ainda é desconhecida e nos eucariotes as regiões dos introns são muito mais extensas que a dos exons! Além deste problema há um segundo problema que também dificulta baixar a estimativa da Entropia do DNA. No DNA há muita informação. Podemos citar que de um único ovo fertilizado (uma célula) nasce um animal inteiro, que no caso dos sapos passa até por metamorfose de girino para sapo.

Pelo que foi visto acima a entropia é a media da quantidade de informação. Quanto maior a entropia maior a quantidade de informação e menos compressível é a mensagem. O DNA possui entropias distintas nos introns e nos exons. Embora se tenha conseguido diminuir bastante a entropia em alguns estudos houve, em outros estudos, uma dificuldade muito grande em se comprimir as proteínas (exons).

O grande desafio na correta estimativa da Entropia do DNA está na compreensão dos mecanismos pelo qual a informação contida no DNA é organizada no próprio DNA. Por outro lado o aspecto biológico tem uma importância maior até que uma baixa entropia para o DNA. O estudo da Entropia do DNA pode ajudar na localização de regiões do DNA que possuam significado biológico (mais uma vez está presente a compreensão dos processos no qual o DNA está envolvido). Talvez a baixa taxa de Entropia do DNA somente seja possível após algum avanço na Engenharia Genética, ou talvez os estudos da Teoria da Informação nas moléculas de DNA ajudem no avanço da Biologia ao encontrar características ainda desconhecidas no DNA.

O estudo da Entropia do DNA envolve além dos conhecimentos de Informática e Biologia Molecular uma boa dose de criatividade. Os artigos que conseguiram as mais baixas taxas de entropia buscaram alternativas não muito convencionais como modelos gramáticos e comparação inexata, o que não é muito usual nos algoritmos para a linguagem natural. Talvez este seja o caminho para se descobrir os caminhos do DNA, a criatividade. Ao se reunir os conhecimentos específicos com a criatividade pode-se descobrir algum pedaço deste grande enigma que é o DNA.

Referências

[1] T.C. Bell, J.G. Cleary e I.H. Witten, *Text compression*, Prentice Hall, 1990.

[2] M. Farach, M. Noordewier, S. Savari, L. Shepp, A. Wyner e J. Ziv, On the entropy of dna: Algorithms and measurements based on memory and rapid convergence, *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM-SIAM, 1994, pp. 48–57.

[3] F. Gewandznajder e S. Linhares, *Biologia hoje*, editora Ática, 1995.

[4] D. Gusfield, *Algorithms on strings, trees, and sequences computer science and computational biology*, Cambridge University Press, 1999.

[5] J.K Lanctot, M Li e E.-H. Yang, Estimanting dna sequence entropy, *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM-SIAM, 2000, pp. 409–418.

[6] D. Loewenstern e P.N. Yianilos, Significantly lower entropy estimates for natural DNA sequence, *Journal of Computational Biology* **6** (1999), no. 1, ??–??

[7] C.G. Nevill-Manning e I.H. Witten, Protein is incompressible, *Proc. Data Compression Conference*, IEEE Press, 1999, Los Alamitos, CA, pp. 257–266.

[8] K. Sayood, *Introduction to data compression*, Morgan Kaufmann, 2000, Second Edition.

Apêndice 1

Um exemplo da implementação da compressão aritmética decimal passo a passo.

Dada a mensagem $ELISAFONSECAHORTA$. e a distribuição de probabilidades abaixo codificar a mensagem levando-se em conta que a precisão do registrador da máquina trabalha com a precisão máxima de 15 algarismos

Código				
A	[0	,	0,15);	
	[0,15	,	0,25);	Espaço em branco
E	[0,25	,	0,35);	
O	[0,35	,	0,45);	
S	[0,45	,	0,55);	
.	[0,55	,	0,60);	Ponto final (fim da mensagem)
F	[0,60	,	0,65);	
I	[0,65	,	0,70);	
R	[0,70	,	0,75);	
T	[0,75	,	0,80);	
C	[0,80	,	0,85);	
L	[0,85	,	0,90);	
N	[0,90	,	0,95);	
H	[0,95	,	1).	

Será feita uma implementação em 12 algarismos.

Toda vez que aparecer o símbolo '*n' 'n' algarismos estarão sendo retirados dos intervalos superior e inferior e transmitidos para o receptor. Na linha de baixo constarão os novos valores já ajustados após a saída dos 'n' algarismos. Após haver a primeira transmissão de algarismo haverá uma linha informando os algarismos transmitidos, para cada novo símbolo codificado na mensagem.

Símbolo Número transmitido	Limite Inferior	Limite Superior	Tamanho do intervalo	
Início	000000000000	1000000000000	1000000000000	
E	250000000000	350000000000	100000000000	
L	335000000000	340000000000	005000000000	*1
	350000000000	400000000000	050000000000	
3				
I	382500000000	385000000000	002500000000	*2
	250000000000	500000000000	250000000000	
338				
S	362500000000	387500000000	025000000000	*1
	625000000000	875000000000	250000000000	
3383				
A	625000000000	662500000000	037500000000	*1
	250000000000	625000000000	375000000000	
33836				
Espaço	306250000000	343750000000	037500000000	*1
	062500000000	437500000000	375000000000	
338363				
F	287500000000	306250000000	018750000000	
338363				
O	294062500000	295937500000	001875000000	*2
	406250000000	593750000000	187500000000	
33836329				
N	575000000000	584375000000	009375000000	*1
	750000000000	843750000000	093750000000	
338363295				
S	792187500000	801562500000	009375000000	
338363295				
E	794531250000	795468750000	000937500000	*2
	453125000000	546875000000	093750000000	
33836329579				
C	528125000000	532812500000	004687500000	*1
	281250000000	328125000000	046875000000	
338363295795				
A	281250000000	288281250000	007031250000	*2
	125000000000	828125000000	703125000000	
33836329579528				
Espaço	230468750000	300781250000	073125000000	
33836329579528				
H	297265625000	300781250000	003515625000	
33836329579528				
O	298496093750	298847656250	000351562500	*3
	496093750000	847656250000	351562500000	
33836329579528298				
R	742187500000	759765625000	017578125000	*1
	421875000000	597656250000	175781250000	
338363295795282987				
T	553710937500	562500000000	008789062500	*1
	537109375000	625000000000	087890625000	
3383632957952829875				
A	537109375000	550292968750	013183593750	*1
	371093750000	502929687500	131835937500	
33836329579528298755				
. Final	443635515625	450195312500	006591796875	*1
	436035156250	501953125000	065917968750	
338363295795282987554				

Neste ponto é necessário fechar a transmissão, colocando um valor único que caracterize. Um critério que pode ser adotado é o critério de enviar o menor número de caracteres. Neste caso uma das opções possíveis seria o 46 já que $43 < 46 < 50$ garantindo assim que o número transmitido está dentro do intervalo exigido pelo final da mensagem. Desta forma o número final ficou: 33836329579528298755446. Antes de lidarmos com o decodificador há uma observação que devemos destacar: no início estávamos utilizando 2 algarismos significativos no final tivemos que utilizar os 12 disponíveis, ou seja: Se a mensagem fosse um pouco maior provavelmente não caberia em 12 algarismos e haveria necessidade de se fazer ajustes. Não é difícil de se fazer, mas não cabe neste apêndice. Agora trataremos do decodificador: O decodificador trabalha da mesma forma que o codificador, apenas com a diferença de os intervalos inferior e superior serem dados pelo codificador. Em função destes intervalos é que são computados os símbolos originais e decodificada a mensagem. Lembramos que para o dado intervalo a porcentagem a ser levada em conta fornece o símbolo na mesma tabela de entrada definida acima. Vejamos como fica o decodificador para a mensagem 33.836.329.579.528.298.755.446.

Limite Inf.	Limite Sup.	Valor Fornecido	Símbolos Decodificados Ajustes
000.000.000	1.000.000.000	338.363.295	Falta tratar 79.528.298.755.446
250.000.000	0.350.000.000	338.363.295	$E \rightarrow$ *Subtrair* 250000000;
000.000.000	0.100.000.000	088.363.295	$\times 10$ e baixa novo número:
000.000.000	1.000.000.000	883.632.957	Falta tratar 9.528.298.755.446
850.000.000	0.900.000.000	883.632.957	$L \rightarrow$ Subtrair 850000000;
000.000.000	0.050.000.000	033.632.957	$\times 10$ e baixa novo número:
000.000.000	0.500.000.000	336.329.579	Falta tratar 528.298.755.446.
325.000.000	0.350.000.000	336.329.579	$I \rightarrow$ Subtrair 325000000
000.000.000	0.025.000.000	011.329.579	$\times 10$ e baixa novo número:
000.000.000	0.250.000.000	113.295.795	Falta tratar 28.298.755.446.
112.500.000	0.137.500.000	113.295.795	$S \rightarrow$ Subtrair 112500000
000.000.000	0.025.000.000	000.795.795	$\times 10$ e baixa novo número:
000.000.000	0.250.000.000	007.957.952	Falta tratar 8.298.755.446.
000.000.000	0.037.500.000	007.957.952	$A \rightarrow$ Nada a subtrair
000.000.000	0.375.000.000	007.957.952	$\times 10$ e baixa novo número:
000.000.000	0.375.000.000	079.579.528	Falta tratar 298.755.446.
056.250.000	0.093.750.000	079.579.528	$\rightarrow$ Subtrair 56250000.
000.000.000	0.037.500.000	023.329.528	$\times 10$ e baixa novo número:
000.000.000	0.375.000.000	233.295.282	Falta tratar 98.755.446.
225.000.000	0.243.750.000	233.295.282	$F \rightarrow$ Subtrair 225000000
000.000.000	0.018.750.000	008.295.282	$\times 10$ e baixa novo número:
000.000.000	0.187.500.000	082.952.829	Falta tratar 8.755.446
065.625.000	0.084.375.000	017.327.829	$O \rightarrow$ Subtrair 65625000
000.000.000	0.018.750.000	017.327.829	$\times 10$ e baixa novo número:
000.000.000	0.187.500.000	173.278.298	Falta tratar 755.446
168.750.000	0.178.125.000	173.278.298	$N \rightarrow$ Subtrair 168750000
000.000.000	0.009.375.000	004.528.298	$\times 100$ e baixam 2 números:
000.000.000	0.937.500.000	452.829.875	Falta tratar 5.446
421.875.000	0.515.625.000	452.829.875	$S \rightarrow$ Subtrair 421875000
000.000.000	0.093.750.000	030.954.875	$\times 10$ e baixa novo número:
000.000.000	0.937.500.000	309.548.755	Falta tratar 446
234.375.000	0.328.125.000	309.548.755	$E \rightarrow$ Subtrair 234375000
000.000.000	0.093.750.000	075.173.755	$\times 10$ e baixa novo número:
000.000.000	0.937.500.000	751.737.554	Falta tratar 46
750.000.000	0.796.875.000	751.737.554	$C \rightarrow$ Subtrair 750000000
000.000.000	0.046.875.000	001.737.554	$\times 10$ e baixa novo número:
000.000.000	0.468.750.000	017.375.544	Falta tratar 6
000.000.000	0.070.312.500	017.375.544	$A \rightarrow$ Nada a subtrair:
000.000.000	0.070.312.500	017.375.544	$\times 10$ e baixa novo número:
000.000.000	0.703.125.000	173.755.446	
105.468.750	0.175.781.250	173.755.446	$\rightarrow$ Subtrair 105468750
000.000.000	0.070.312.500	068.286.696	$\times 10$:
000.000.000	0.703.125.000	682.866.960	
667.968.750	0.703.125.000	682.866.960	$H \rightarrow$ Subtrair 667968750
000.000.000	0.035.156.250	014.898.210	$\times 10$:
000.000.000	0.351.562.500	148.982.100	
123.046.875	0.158.203.125	148.982.100	$O \rightarrow$ Subtrair 123046875
000.000.000	0.035.156.250	025.935.225	$\times 10$:
000.000.000	0.351.562.500	259.352.250	
246.093.750	0.263.671.875	259.352.250	$R \rightarrow$ Subtrair 246093750
000.000.000	0.017.578.125	013.258.500	$\times 100$:
000.000.000	1.757.812.500	1.325.850.000	
1.318.359.375	1.406.250.000	1.325.850.000	$T \rightarrow$ Subtrair 1318359375
000.000.000	0.087.890.625	007.490.625	$\times 100$
000.000.000	8.789.062.500	749.062.500	
000.000.000	1.318.359.375	749.062.500	$A \rightarrow$ Nada a subtrair.

Neste ponto a razão do valor fornecido pelo tamanho do intervalo assume 56,81de final da mensagem está em [0,55 , 0,60)). Foi possível tratar a recepção de forma distinta da transmissão (utilizando entre 9 e 10 algarismos) pois não foi necessário tratar os bytes finais, pelo algoritmo implementado. Porém ficou claro que deve-se haver uma política de truncamento (tanto no transmissor quanto no receptor) para que não ocorra esta necessidade de aumento crescente do número de algarismos para tratar a mensagem. Isto está abordado com detalhes no texto principal.