A Parallel, High-Throughput Framework for Discovery of DNA Motifs

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Kyle W. Kurz

June 2010

This thesis titled

A Parallel, High-Throughput Framework for Discovery of DNA Motifs

by

KYLE W. KURZ

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Lonnie R. Welch

Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

KURZ, KYLE W., M.S., June 2010, Computer Science

A Parallel, High-Throughput Framework for Discovery of DNA Motifs (315 pp.)

Director of Thesis: Lonnie R. Welch

The search for genomic information has just begun. New genomes are sequenced daily, and each brings new challenges and knowledge to the scientific table that must be carefully mined and studied to glean out every possible bit of information. The amount of data created during genomic sequencing is simply too great for researchers to handle, creating a need for computational tools capable of processing the genomic input and analyzing it for information. The area of bioinformatics focuses on this combination of computer science and biology, bringing useful software applications to the table in an effort to ease the workload of biologists.

One specific area of interest to biological researchers is the study of DNA words or motifs as they relate to gene regulation. These regulatory elements may be transcription factor binding sites (TFBS), which bind RNA polymerase II to the DNA strand, or enhancer/silencer sequences that up- and down-regulate transcription of the gene to which they are related by binding specific proteins. Many tools such as Weeder [43], WordSpy [65] and YMF [55] are currently available for the study of over- and under-represented words in a DNA sequence, a trait which is believed to useful in identification of these regulatory elements. These tools all perform similar tasks by enumerating all words, or substrings, found in their input, then scoring and ranking these resulting words for presentation to the user. Optionally, many tools also cluster groups of words together to form degenerate motifs which allow for evolutionary and environmental variation in the binding site.

The Open Word Enumeration Framework (OWEF), presented in this thesis, provides a new framework on which DNA word enumeration tools can be built. The OWEF

framework provides a set of abstract base classes representing the core stages of a word enumeration tool and defines a set of standard interfaces for each stage, allowing multiple algorithmic implementations of these base classes to co-exist and be selected individually at runtime.

In addition to providing a level of abstraction that allows for simpler development, the framework also provides a scalable solution to alleviate memory bottlenecks. The framework contains skeleton code for both a shared memory implementation, providing fast analysis on single-node, multiprocessor systems, and a distributed memory solution, which splits the tasks among several networked nodes to provide a large amount of accessible main memory to the application.

In summary, the OWEF framework is useful as a development tool by providing a set of interfaces and methods to allow developers to focus on specific aspects of the algorithms they are designing, while also providing a standardized, flexible interface to researchers, eliminating the need for specialized tools and providing a general-purpose toolkit for DNA word enumeration tasks.

Approved: _____

Lonnie R. Welch

Professor of Electrical Engineering and Computer Science

For Becky.

I love you.

Acknowledgements

Jens Lichtenberg, thanks for pulling me through; Dan Evans, Xiaoyu Liang, Paul Burns, Rami Al-Ouran, Lee Nau, Lev Neiman, Alper Yilmaz, Matt Wiley, thanks for working with me; Frank Drews and Lonnie Welch, thanks for giving me the opportunity to get here.

Jens Lichetenberg, Xiaoyu Liang, Josh Welch, Rami Al-Ouran, Lev Neiman and Lee Nau, thanks for your work on the WordSeeker and OpenMotif projects and help maintaining and improving the OWEF codebase.

Johnny Barton, Wes Woodall, Victor Rasgaitis, Tim Wasserman, thanks for playing music with me to give my mind a break. I truly enjoyed making music with all of you and wish we could do that forever. Meg Alexander, thanks for your constant support of my artistic side projects.

Andrew Costa, Paul Risler, Destry Flick, Matt Barnett, Ricky Chilcott, Josh Schnacke, Andy Vogt, thanks for the nights of ultimate frisbee, Bible study and alien shooting. You guys have become some of my closest friends, I will miss spending nights on the field when I move...

Thanks to Chris Tyo, Ken and Karen Kurz, and Jens Lichtenberg for proofreading.

Table of Contents

List of Tables

List of Figures

List of Listings

List of Acronyms

API - Application Programming Interface
DNA - Deoxyribnucleic Acid
HPC - High Performance Computing
MITRA - Mismatch Tree Algorithm
MPI - Message Passing Interface
OBF - Open Bioinformatics Foundation
OSC - Ohio Supercomputer Center
OWEF - Open Word Enumeration Framework
PWM - Position Weight Matrix
RAM - Random Access Memory
RNA - Ribonucleic Acid
RPC - Remote Procedure Call
SMD - Substring Minimal Distance
SMS - Simple Motif Search
TSS - Transcription Start Site
UTR - Untranslated Region

1 Introduction

In the post-"Human Genome Project" [64] world, bioinformatics research has blossomed into a thriving area of research as many groups, both academic and corporate, attempt to mine the genomes of thousands of organisms for valuable information that could lead to tremendously important scientific discoveries. New tools and algorithms are constantly being developed to aid these research goals and simplify the discovery process. This proliferation of new software can easily leave users feeling abandoned or overwhelmed as their tool of choice becomes obsolete or a new tool comes along that can be shown as better on certain datasets. Often, users can miss newer, high-quality algorithms that could process their data more effectively because they choose to stay with a familiar interface they know and understand. This tendency of users to favor an application they have been using for some time helped define and shape the design of the Open Word Enumeration Framework, or OWEF, with the goal of being "the last tool you ever need to learn" for word enumeration tasks by providing a consistent front-end user experience while allowing developers to build in new and useful algorithms for common tasks seamlessly and easily.

1.1 Background

For several years, bioinformatics research groups at universities and companies have been churning out hundreds of task-specific tools written in various languages and appealing to certain minor groups of the research community. Often, these tools are made available to the outside world through open-source licenses or official releases and are slowly adopted into workflows as biology researchers search for an ideal chain of applications that can work together to perform the analysis they need to understand the data produced during experimentation. As datasets grow in size and groups abandon development, it is easy to imagine the day when a favorite program is incapable of

handling the data provided, and the search for a new favorite must begin, and adoption of this new tool often comes with a steep learning curve. As the performance of computing hardware moves from single core processors and low main memory (RAM) increases, new tools must integrate paralellization and handle large datasets to remain relevant in the research community.

1.1.1 Many Unique Tools

A special subset of bioinformatics tools contains applications for the enumeration of DNA words. DNA words are strings of characters representing nucleotides, provided to analysis tools as sequences, which could alternatively be considered as very long words. Typically, biology researchers are interested in shorter words, which may be represented as substrings of the provided sequences, and statistics about how often those words occur compared to how many times they would be expected to occur via a statistical model. Words that are either highly over- or under-represented are then selected as a starting point for further research into their function within the sequence, chromosome or genome.

WordSpy [65], Weeder [43] and R'MES [27] are only three of the many tools available for DNA word enumeration studies. In addition to tools like YMF [55] and Seeder [19], it is possible to build a tool from various existing tools by writing intermediate scripts to transform the output of one tool into the appropriate input for another tool. Frameworks like BioPerl [60] provide high-quality general purpose algorithms for bioinformatics, such as parsing input files or querying databases for information. Similar toolkits exist in the Java (BioJava [28]) and Python (Biopython [11]) languages. These tools, among others, are discussed in the literature review (Chapter 2).

1.1.2 Awash in Data

As genomic sequencing costs lower, databases are rapidly being created with the full genome sequence of many organisms. Where researchers once had access to only a select

few model organisms, they may now take their pick of specific sub-species, or study a newly sequenced organism that no one else has studied yet. This is truly an exciting time to be working in genomic studies, but the sheer amount of data available can be overwhelming. Datasets for *Arabidopsis thaliana*, for example, provide not only the entire genome, but also seven other files for the core, distal and proximal promoters, the gene coding sequences, the intronic regions and the 3' and 5' untranslated regions around the genes. Eventually, each of these datasets will be available for every sequenced genome, in addition to the interesting custom datasets created during biological experiments. Just tracking these files, let alone analyzing them, will become a daunting task.

1.1.3 The WordSeeker Project

One of the tools to come out of the bioinformatics lab at Ohio University is the WordSeeker tool, a word enumeration-based statistical analysis package for DNA word and motif discovery. Originally built as a set of tools that communicated via flat files, the redesign of the tool as a single executable sparked the initial concept of the OWEF framework. The communication between each stage of the pipeline that was defined during implementation of early WordSeeker versions was examined and a small set of necessary interfaces were identified which could provide an abstract representation of the various stages, allowing the incorporation of many algorithms that would normally require recompilation of the codebase, at minimum, and separate tools, in the worst case.

1.1.4 Proposing a Solution

Solving these problems with a single approach would be nearly impossible. Instead, a more general purpose solution is needed to allow fine-grained user control over the algorithms used to perform each task in the analysis of their dataset. The OWEF framework provides more than a single solution, as the framework itself does not contain the functionality necessary to process data. Instead, the framework provides

communication protocols for the most common stages used in DNA sequence analysis, allowing many algorithms to be included in the tools built on the framework. If a specific job requires fast access times to words in the input file, a linear-time data structure can be implemented/selected. If the user needs to know what words are within a Hamming distance of the top scored words in the dataset, a Hamming-based clustering method can be added to the framework.

1.2 Motivation

Several motivating factors drove the development of the framework described here, the most important being the ability to analyze not only small datasets, but full genomes while retaining a consistent interface for users. The former requirement also brings requirements for paralleism and scalability, as many full genomes will easily require more main memory than a single computational node can provide. The combination of these goals brings significant benefit to members of the bioinformatics community by providing biological researchers with a powerful, easy-to-use tool and providing developers with a set of algorithm interfaces that allows them to easily implement new algorithms for one of the predefined stages or use the existing algorithms in conjunction with a new stage to avoid building an entirely new tool to support their new method.

1.2.1 High-throughput and High-scalability

The Weeder [43] online tool limits users to twenty thousand characters in the input data. R'MES [27] was tested in the Ohio University bioinformatics lab and is incapable of handling word lengths longer than thirteen for datasets as small as the *Escherichia coli* genome (~4 MB). These tools could not be considered high-throughput, and certainly do not scale with respect to input size. A primary goal in the creation of the OWEF framework was to create a toolkit capable of handling almost any input dataset and processing it to provide useful results to the user within a few minutes or hours, instead of

days. To realize this goal, the framework was designed to operate on two distinct computational platforms; a single-node version was implemented to run on personal computers and lab machines, and a multi-node version was created to allow processing in a distributed High Performance Computing (HPC) environment capable of processing exteremely large datasets up to long word lengths.

1.2.2 Consistency

Every new algorithm deserves its own tool. This has been the general consensus of the bioinformatics developer community to this point. Unfortunately, this mindset hinders users by forcing them to learn a new interface each time they need a new tool for a job. The OWEF framework provides both developers and users with a consistent interface, virtually eliminating the learning curve of adopting new tools based on the framework. The framework itself could be thought of as a stand-alone tool, with the various algorithms providing user-specified options, allowing the user the flexibility of selecting the appropriate data structure to process their input. If, for instance, the default data structure is too large to fit in main memory, the user may choose to use a more space-efficient algorithm, at the cost of longer runtimes.

Developers, on the other hand, do not need to reinvent the wheel when implementing new algorithms. If a new scoring model is designed by a lab, the use of the framework will allow the use of the existing data structures and word clustering methods, immensely lightening the load on the development team. Additionally, the new model can then be easily compared to existing models to show its applicability to the problem at hand, providing instant validation with a strong experimental setup, as the other algorithms in the comparison are static. Chapter 3 provides detailed information about the layout of the framework and how developers can interface with the existing code.

1.2.3 Full-genome Analysis

A human's personal genome, when stored in the cell, is compressed to fit within the nucleus of each and every 10 µm. The human genome, when stored as a string of characters in a file, is larger than 3 GB, with other genomes, such as maize, requiring even more storage space. Many consumer computers have less main memory than this, making analysis of these large datasets extremely difficult or impossible. Those that do have enough memory for the input, will quickly run out of space as enumerative bioinformatics tools begin to process and build data structures capable of holding information about every unique substring, or word, present in the input data. Tools such as YMF [55] restrict the user to ten thousand characters, far from the three billion required for the human genome!

The cost of main memory has decreased drastically in recent years, but hardware upgrades can only provide limited increases to input size limits. The next logical step to increase available main memory after maximizing the RAM on a single machine is to move to parallel algorithms that allow several computational nodes to act as a unit to process disjoint parts of the input. With this model, when more memory is necessary, more computers can be added to alleviate the problem.

An extension of scalability, the analysis of whole genomes is a highly active area of research, with many complex tradeoffs involved in keeping computation runtimes down and avoiding thrashing, which occurs when a task attempts to use more main memory than available and the operating system must begin swapping information to the hard disk, which will eventually bring the system down and render computation impossible. Genomic analysis was identified as a goal for the OWEF framework during the early planning stages, allowing for the design of both single-node and distributed systems within the framework. These designs are discussed in greater detail in Chapter 4.

2 The State of Bioinformatics Software

The framework described in this document provides an abstraction for the basic stages of a DNA word enumeration tool. Applications that fit into the category of enumerative must be able to systematically return every word of a given length and provide basic statistics about it, such as how many times the word occurred in the input and in how many of the input sequences the word was found. This basic information is then used by scoring models to caclulate valuable statistics about the word's importance in the dataset, giving researchers a way to sort the words and find a starting point for continued research.

2.1 Current Word Enumeration Tools

The world of bioinformatics research is full of tools vying for dominance as the go-to tool for mining data from DNA sequence files. This competition creates both a positive and negative effect on the development community. The number of groups working on these algorithms means a constant drive for improvement and discovery. On the other hand, since all the tools have essentially the same goal, there is a tremendous amount of redundancy among the tools.

The following subsections describe many of the most popular enumeration tools employing a variety of approaches, both in their choice of data structure for storage and their reported statistics. A discussion of the common tasks of all these tools then provides a strong motivation for the creation of the OWEF toolkit, presented in Chapter 3.

2.1.1 MITRA

The MITRA tool [18], which is an acronym for MIsmatch TRee Algorithm, performs word and motif discovery on an input dataset using a combination of a tree data structure, similar to suffix trees and radix trees, and a graph to identify a word and the set of words

that are within a distance, d, of the seed pattern. As an over-simplification, setting d equal to zero will result in scoring and analysis of words with no variation. The MITRA data structure is very space efficient, as it does not retain the entire tree in memory at any given time. When a word does not meet the parameters of the search algorithm, the subtree from that node is removed, saving space that will never be used. MITRA is based on the SDA algorithm proposed by Waterman et. al [66], with a modification that eliminates searching all possible words by seeding the algorithm with words that are known to exist in the dataset.

2.1.2 REPuter

Using a suffix tree as its storage system, the REPuter tool [31] claims a running time of $O(n + m)$ and space requirement of $O(n)$, where n is the length of the input and m is the number of elements output by the tool. This algorithm is an optimization of the general suffix tree concept [38] and does not contain any scoring, but is presented as a possible database for further development. Also included is a tool that visualizes the information output by REPuter as a ”repeat graph” [31].

2.1.3 R’MES

Developed by Schbath and Hoebeke, the R’MES tool [27, 53] uses a hash table to store the database of words found in the input, then uses Markov modeling statistics to identify important words. The R’MES tool is limited in its processing capablities, especially with respect to word length. When scoring words, the tool creates every possible word of the requested length and then polls the hash table for the number of occurrences of the word, resulting in many searches for words that do not appear. The statistics package contained in R’MES is extremely powerful, however, and includes Poisson and other models in addition to the default Markov model.

2.1.4 Seeder

Seeder works with short words and computes statistics based on the occurrence of similar words, identified by the SMD, or Substring Minimal Distance [19]. Using a radix tree storage approach, Seeder stores indices on each node identifying other words contained in the data structure that are within a specified Hamming distance of the word indicated by the node. The Seeder algorithm computes a distribution function for each word based on its distance from a randomly selected word in the background model. To compute a position weight matrix for the word, the sum of the word distribution is computed using the SMDs listed on the node and the n-fold self-convolution of $g_w(y)$ [19] to compute the values of each index of the PWM. Seeder is typically used to find motifs, as indicated by the computation of the PWM, however setting the Hamming distance to zero will yield the special case where the motif is simply a static word, with no mutations allowed.

2.1.5 SMS

Like Seeder, the SMS, or Simple Motif Search, algorithm is not in itself a full word enumeration tool. The SMS algorithm uses a radix sorting technique to report words and their number of occurrences [46]. The authors claim a worst case running time of $\Omega(n^2)$, and an expected run time of $O(n\log n)$. The main drawback of this approach is the use of simple arrays as a storage system. For longer word lengths, the space wasted by explicitly storing all characters of each word will quickly overload the main memory of the system. If the arrays can be stored in main memory, however, this algorithm will execute quickly, as radix sort can sort the arrays in linear time.

2.1.6 Speller

The Speller algorithm builds a suffix tree using the input sequence(s) and then performs a tree-traversal to ”spell” out all words of a requested length [51]. This approach is also capable of containing the data structure in linear space with respect to the length of the input. For approximate string matching, suffix trees provide possibly the best combination of fast access to data and (relatively) small memory footprint. When mismatches are not considered, other structures may perform better in one complexity or the other.

2.1.7 TEIRESIAS

The TEIRESIAS algorithm [49] was developed at IBM as a way to identify maximal patterns in DNA data. That is, the algorithm has a threshold for minimum wordlength, but if a longer word occurs exactly as many times as a shorter word, and the shorter word is a substring of the longer, TEIRESIAS will only report the longer word, and the occurrence of the shorter word can be inferred. The TEIRESIAS algorithm is also capable of reporting degenerate motifs, and its input includes the maximum number of wildcard characters that may be contained in a reported pattern.

2.1.8 Verbumculus

Another suffix tree based tool, Verbumculus performs statistical analysis on a subset of the words contained in the input by computing the expected occurrence value and variance of each word and then applying a normalization function to the words to determine if they are valuable enough to be presented to the user as output [4]. Like the TEIRESIAS algorithm, the Verbumculus tool limits its output by checking to see if shorter words are fully contained within longer words and therefore unnecessary. Verbumculus

also includes a visualization tool available via their web version that visualizes the created output for the top 100 scored words.

2.1.9 Weeder

Described in [42, 43], the Weeder tool is publicly available on the Internet, and available as source code from the authors' website. The Weeder algorithm [42] builds a suffix tree and limits the output to only words with a specific number of occurrences. The output of Weeder is sorted by the Z-score that is computed by the tool, which is a measure of how many more times the word occurred than expected. Using the suffix tree's ability to easily identify mismatched strings, the authors present statistics showing that the algorithm performs well on motif-finding exercises, where the length and number of mismatch positions is known.

2.1.10 WINNOWER

The WINNOWER algorithm is unique in this set of tools, as it constructs a graph for the input [45] and searches the graph for cliques to allow for mismatches between words. The algorithm works by first looking at the size of the graph that is created for each pattern. A bad pattern is indicated by a very large graph, and an empty graph indicates that no pattern was found. A relatively small graph that contains one clique of appropriate size is considered to be a valid pattern which is subsequently scored and output to the user. It should be noted that WINNOWER was only tested in [45] for sequences of length 100 through 1000, which could prove to be a serious limitation of the algorithm.

2.1.11 WordSpy

WordSpy utilizes a hash table to store what the authors describe as the ”stegoscript” for the input data [65]. The tool learns a dictionary of all the words of a length, then moves on to the next word length and builds a new dictionary until it arrives at the word

length desired. The dictionaries of the smaller words are then used to score the words with a Z-score. The resulting output is scored and ranked by this score and a threshold may be applied to limit the output of the tool. WordSpy is available online with input size limitations or as a stand-alone tool for large dataset processing.

2.1.12 Yeast Motif Finder

The Yeast Motif Finder, or YMF tool uses an array that is hashed on the ordering of the alphabet, like the node pointers within a radix trie approach, where A maps to 1, C to 2, etc [56]. YMF counts the nucleotide frequencies and then enumerates every word of the requested length, with or without degeneracy, and then counts and scores the enumerated words. Because the tool enumerates all possible words, its space and time complexity limit its upper boundaries from operating on long words, although it has been shown to be useful on data which the tool is able to process [7, 55, 57].

2.1.13 On Common Ground

With the right parameter settings, each of these algorithms and tools will output all words contained in the dataset to their respective scoring stages. It is important to note that, although most of the tools are designed to work with degenerate motifs, a pure word is simply a special case of a motif with no inconsistencies at any position. The general method of building a database, querying the database for words or motifs and then reporting the results to the user is the basis for the OWEF framework. Since all tools must perform certain steps, the standardization of these simple building blocks can provide a stability while allowing tremendous flexibility. A similar idea has been adopted by operating systems developers, where much of the low-level functionality is incomplete in the basic system. The device manufacturers are required to provide drivers to interface between their components and the operating system. The same can be said for developers using the OWEF tool. While the tool takes care of many general maintenance tasks, the

core functionality of word enumeration and scoring rests in the various implemented techniques that communicate through the standard interfaces built into the framework.

2.2 Current Open-Source Bioinformatics Frameworks

An alternative to a single-purpose toolkits, several groups have been working to provide general purpose, modular frameworks that allow developers to reuse basic code, such as reading in files in standard formats, querying databases and performing visualizations. Developers can then write scripts to connect these modular pieces of code and create their own code. Unfortunately, these frameworks provide only one version of these modules, and defining how to use them is left to the developer, which could lead to inefficient communication and use.

2.2.1 Bio++

Published in [17], the Bio++ package was part of a larger effort in the bioinformatics community to provide commonly used functionality within a set of libraries in each major programming language. Possibly due to the rarity to which C++ is used by biologists, or possibly because of lack of adoption by the Open Bioinformatics Foundation (OBF), which curates the hosting and repositories for the other packages listed below, the Bio++ project has ceased development and was last updated in 2006, making adoption futile without a large demand from the user community. Bio++ provided containers for reading and accessing sequences, phylogeny tools and general utilities for performing common bioinformatics research calculations. When published, the libraries were in their infancy and they unfortunately never had a chance to mature to the level of the other Bio(*) frameworks.

2.2.2 BioJava

The BioJava framework provides a set of high level API's (Application Programming Interfaces) for developer use, including the standard sequence manipulation objects common to the other frameworks curated by the OBF, along with alphabet abstractions and, perhaps the most valuable, an API for serialization of classes to be stored in users' databases [28]. Many other interfaces are provided, and more will be added as development continues. Like all OBF projects, BioJava is open-source.

2.2.3 BioPerl

Perhaps the most well-known and used of the OBF frameworks, BioPerl is an extremely mature and robust package used in many open-source bioinformatics projects [60]. Providing functionality for tasks as varied as querying remote databases for data to performing sequence alignment, the BioPerl framework has hundreds of bioinformatics-specific modules that can be used in combination to build a tool for most tasks with almost no additional code requirements. The popularity of the BioPerl tools can likely be attributed to the Perl language's popularity in biological circles, as it provides a good tradeoff between ease-of-use and processing power. Perl's flexibility in string processing is unmatched, also making it an ideal language for studying the language of biological communication.

2.2.4 Biopython

Biopython contains many of the same API's as BioPerl and BioJava, written in Python [11]. Python is a terrific language for middleware applications between other tools, as it can easily communicate with native code, while remaining a high-level programming language. The Biopython framework brings this convenience to the bioinformatics community, allowing the implemenation of longer, more valuable execution pipelines.

2.2.5 BioRuby

BioRuby's claim to fame is its BLAST [3] parsing capability, which the author claims is "about 10 times faster than BioPerl" [21]. Many of BioRuby's API's revolve around interfacing with tools over the web, an area where the Ruby language shines. Additionally, BioRuby provides sequence input, output and manipulation through data structures objects, as well as strong database connection with BioSQL.

2.2.6 BioSQL

While there is no official paper announcing BioSQL, the application notes of BioJava [28], BioPerl [60], Biopython [11] and BioRuby [21] all make note of their ability to interface with a BioSQL database. The four groups working on the various Bio(*) projects came together to create a standard schema that would allow any of the tools to read in objects from the database, even if the objects were placed in the database through a tool written in another language. This allows easy communication between instances of various bioinformatics programs and provides a reliable storage system for data.

2.3 Combining Ideas

It is possible to combine some of the best aspects of each of these enumeration tools with the idea of creating a modular framework. By identifying standard communication interfaces between components similar to the BioPerl modules, and implementing the underyling data strucures and scoring models of each of the tools in a way that they can use these standard interfaces, a new type of framework can emerge that is extremely flexible while eliminating redundancy of programming and providing a simplification of the development path. To this end, the OWEF framework was designed and built.

3 The Open Word Enumeration Framework

The Open Word Enumeration Framework (OWEF) is an open-source toolkit designed with both developers and users in mind. This chapter describes the high-level outline of the framework and how it can aid in rapid development while providing a consistent long-term interface for users. Additionally, the main function is described in detail to set the stage for following sections detailing the individual stage parent classes. These parent classes are abstract representations of the concepts involved in each stage, and must be overridden by child classes to allow the framework to process data.

3.1 High-Level Design and Architecture

Delegation of tasks has been an important part of this project from the beginning. C++ allows high levels of abstraction, which can be used to create a modular toolkit capable of incorporating many similar algorithms as child classes of a simplified parent class. This goal led to the creation of a control class and several stage-specific parent classes to separate the work involved in processing an entire job into blocks of related tasks. Three major stages that make up the core functionality of the framework were identified: database creation, word scoring and word clustering. In addition, other interesting tasks were grouped into a set of optional post-processing stages: word distribution and scatter plot analysis.

There are three primary parent classes in the framework, along with a higher-level delegation class and multiple child classes for each parent. An object of the delegation class is created for each job run on the framework. This object contains pointers to an object of each of the parent classes and, based on the input arguments provided by the user, instantiates an object of a specific child for each of the parent pointers. This is possible due to C++'s ability to pass a child where a parent is expected. In addition to the

abstract parent classes, the optional post-processing stage for word distribution analysis has been implemented as a separate class to facilitate its development.

During preliminary testing, it was determined that the WordSeeker tool would run out of main memory long before finishing analysis of many genomes when running on only a single computational node. With scalability being an important requirement of the tool, a multi-node solution became part of the design goal. Through the use of compile-time specific code declarations, the tool is able to run in either a distributed or single-node environment with only a simple change to the make arguments. This allows an extremely portable codebase that can be run on any machine with a GNU g++ compiler, but also runs on distributed systems that have an MPI compiler installed.

The basic layout of the framework shows that each parent class can have multiple children, the use of which may be chosen at runtime by the user, depending on the job requirements (Figure 3.1). The number of children each parent may have is limited only by the number of derivative classes implemented. Once the appropriate child classes are selected, job execution utilizes standard interfaces built into each of the parent class definitions to perform job duties. These interfaces remove the need for algorithm-specific knowledge during development by prescribing a specific predictable result independent of the underlying algorithm. For example, the DataStructure getCount function requires a string as its input and must always return the number of unique occurrences of the string provided in the query within the original input file as an integer, regardless of whether the information is stored in a Radix Trie, Suffix Tree or not stored and computed on-the-fly by some alternative database class.

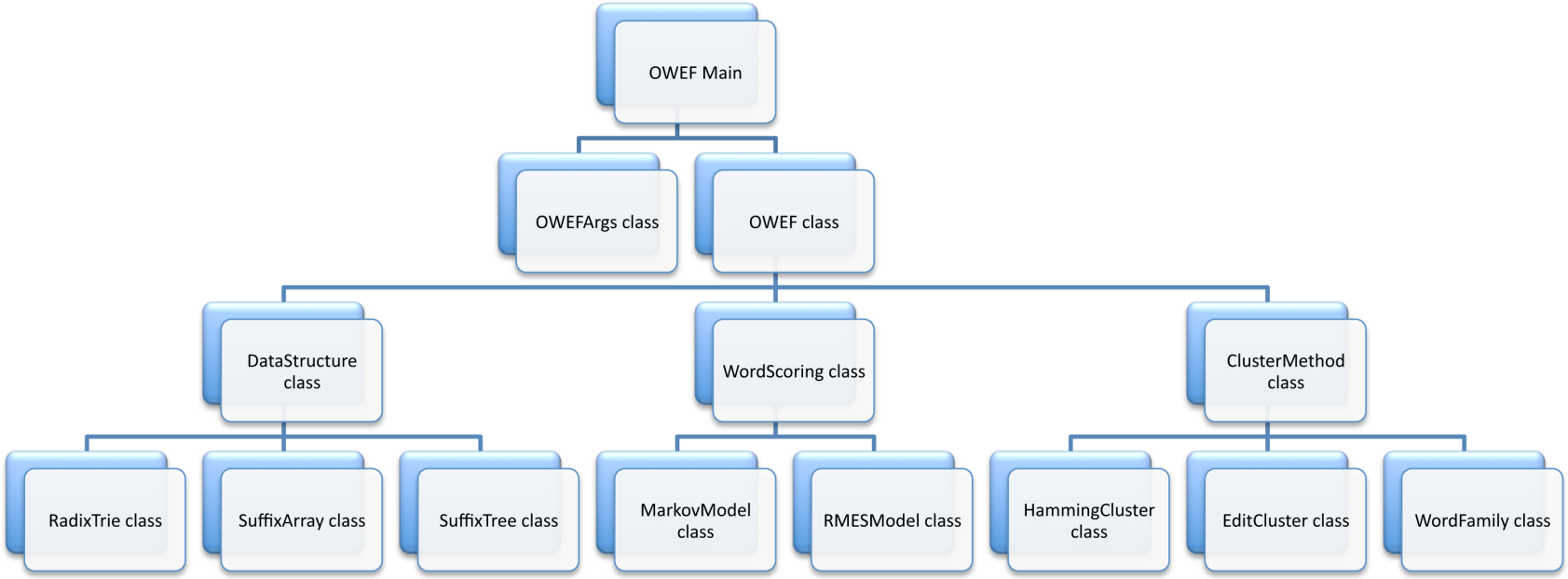

Figure 3.1: Main Class Heirarchy of the OWEF Tool

3.2 Software Design

The development of the tool followed an iterative [32] software design process, as the tool was built as a replacement to an existing toolkit and needed to grow as the needs of the users grew. Preliminary revisions of many stages were highly inefficient, with focus primarily on functionality and further iterations on improving both memory and computational efficiency. Following initial completion of the basic pipeline, each time a new bottleneck was discovered, an analysis of that stage's execution path was performed to identify problem areas within the stage. The first children of each parent were based on pre-existing implementations of the radix trie, markov model and edit/hamming distance clustering algorithms. The initial focus was on correct logical execution of sequential algorithms with verification provided through comparison to existing implementations and naive, brute-force test algorithms. Once a solid sequential algorithm was in place, profiling was performed using the gprof [22] profiling suite to find inefficient sections of code, which were then prioritized and optimized individually, first for memory footprint, and secondly for improved time performance. When an execution time improvement would negatively affect the memory footprint, preference was always given to a lower memory footprint. Memory footprint took precedence over execution times because too large a footprint could potentially lead to system thrashing, which would in turn eliminate any gains made in execution time optimizations.

As developers work on new classes for a given stage, they may think of the other stages as black boxes of code. There is little need to understand, for example, how the DataStructure classes store the information about the words in the input file, only that the information is available through an iterator and that certain information about each word may be obtained through the use of the standard interfaces that can then be used in the new class as needed. Developers may then choose to perform any stage-specific optimizations desired without affecting other stages, so long as the standard interfaces

necessary for the stage in development remain in place. These optimizations could be as simple as keeping a vectorized list of the top ”N” words scored by a WordScoring class child, rather than needing to recalculate the top words when the getSeeds function is called. This simplicity of design is the realization of one of the primary goals for the framework, as it allows developers to focus entirely on their latest technique without designing an entirely new tool.

3.3 The Main Function

The main function for this project, contained in the file OWEFMain.cpp, has the primary task of accepting command line arguments and creating two important class objects that control the flow of the rest of the pipeline. The first object that is created is of the OWEFArgs class, which stores the argument list for the job as provided by the user and each class object created further down the pipeline receives a pointer to this object to identify the optional tasks to be performed in this specific job. The second object is of the class OWEF, which is the high-level delegation class that creates job-specific objects for each stage of the pipeline.

A secondary task of the main function is to make calls to final post-processing stages and to gather the output files into a single directory for the user. Several of the visual data representations available in the pipeline are not complex enough to require their own class, rather they are simple shell scripts that read in the output files from the WordScoring object and create graphics to show the data in an easy-to-understand way. The four scatterplots available represent the distribution of word scores for two of the primary ranking scores in the framework, the p-values of the enumerated words and, if applicable, the estimated rank of the words not contained within the input file. The ranking scores used are the $O * ln(\frac{O}{E})$ score, which is the number of occurrences multiplied by the natural logarithm of the number of occurrences divided by the expected number of occurrences,

and the $S * ln(\frac{S}{E_s})$ score, which is the number of unique sequence occurrences multiplied by the natural logarithm of the number of unique sequence occurrences divided by the expected number of unique sequence occurrences. These scores weight the ranking by total over-representation and high sequence coverage, respectively. High sequence coverage is interesting in datasets with many sequences known to be co-related. Often, a large percentage of words will have similar scores, and a few words will have extremely high or low relative scores. These outliers are targeted for further analysis, as they are statistically relevant. If p-values are calculated for a job, a scatterplot of these p-values may be created to show the statistical validity of the words being analyzed. Finally, users may choose to list nullomers, or unwords, during a run. These nullomers are all words of a specific length that are not found in the input dataset. These words are often statistically interesting as they tend to be under-represented, in other words, they are expected to occur in the dataset, but, for some reason, do not. Evolutionary constraints are often attributed to the elimination of these words, making them extremely interesting in biological research. A graphical representation of these nullomers' expected occurrences in the input set can, again, help identify outliers for further research.

Another critical task within the main function in the distributed system is the spawning and gathering of MPI processes and creation of a map of which host is processing which prefix, which is used heavily in the OWEF class. This distributed functionality is discussed in more detail in Chapter 4. Additionally, the main function performs general job setup and cleanup to ensure reliable jobs without memory leaks. The general execution logic of the main function is shown in Listing 3.1.

```
if( No command line flags)
   Prompt user on job options iteratively
else {
   for(i=1:number of flags){
      Determine which flag is being processed
      Populate appropriate variable to send to OWEFArgs object
   }
}
Set up the prefix & minimum word length variables
Create OWEFArgs object with initialized variables

//If this job is being run in a distributed environment
Set up MPI interprocess communication
if(rank==0)
   Create host map to identify process responsibilities
Send the map to all other ranks

//Do the work of this job
Create OWEF object
if(User desires scatterplots)
   Run any necessary scatterplots
Create results directory
Move all job output files into results directory

//If this job is being run in a distributed environment
Barrier to wait on all processes to finish
if(rank!=0)
   exit
return EXIT_SUCCESS //0
```

Listing 3.1: General Execution Logic of OWEF Main

3.4 The Primary OWEF Classes

The following subsections describe each of the main classes in the framework that are used to build pipelines. The OWEF class provides logical control of the entire framework and selects the appropriate child classes from the available implementations. The DataStructure, WordScoring and ClusterMethod classes provide a high-level definition of the important functions necessary for proper execution and the various post processing tools can significantly enhance the value of the output to the user.

3.4.1 The OWEF Class

In the previous section, the main function for the framework was descibed in detail along with its creation of an argument list in the form of an object of the OWEFArgs class that is passed on to each subsequent stage of the pipeline. This chapter picks up at the next stage of the execution logic and describes the main control class, OWEF.

3.4.1.1 Primary Interfaces and Design

The OWEF class's primary job is task delegation. Based on the user's input arguments, the OWEF class begins the process of analyzing the input data. The OWEF class itself contains little code, with only a constructor, destructor and a thread-safe post-processing function, along with a list of pointer variables for each stage of the pipeline as members. The constructor creates an object for each stage pointer, which in turn causes the execution of each stage as their individual constructors are called. The OWEF class uses the OWEFArgs pointer provided through the constructor to determine which child class object is to be created for each stage pointer. The OWEF class's basic execution follows the following pseudocode (Listing 3.2):

```
if(enumerating words)  //should always be true
   Create DataStructure child object
if(scoring words)  //almost always true
   Create WordScoring child object
Spawn threads
for(all threads)
   Call ThreadStartup2 function
```

Listing 3.2: General Execution Logic of OWEF Class

ThreadStartup2 is a special function defined as an external ”C” function within the framework to allow multiple threads to execute within the C++ code using the pthread_create [9] function. Prior to calling ThreadStartup2, a pointer to the special C++ variable (this), a copy of the OWEFArgs pointer that provides the input arguments and a unique integer are packaged into a struct to allow continued execution following thread startup. Since pthread_create requires a void pointer, a pointer to the struct is recast as void and sent to ThreadStartup2. Upon entering ThreadStartup2, the void pointer is recast as a pointer to the struct and using the pointer to the original object, (this), a call is made to the process function to return execution to the framework within the context of each thread. The process function proceeds as follows (Listing 3.3):

```
if(clustering)
   Create WordClustering child object
if(computing word distributions)
   Create WordDistribution object
```

Listing 3.3: General Execution Logic of the Process function

Upon return from the process function, control is returned to the main function for job finalization. Multiple constructors of each object allow the OWEF class to call an

initialized constructor that automatically calls the processing functions for that stage's tasks without explicit calls from the control class. Each object is only initialized after any precursor stages required are finished; for example, the WordScoring object is not created until the DataStructure object has returned from its constructor, indicating that the database is fully formed and the WordScoring stage may proceed. The process function takes advantage of the full independence of the WordClustering and WordDistribution classes to allow them to process in parallel. This independence is possible because none of these classes require communication with each other, they only require access to the WordScoring and DataStructure objects.

3.4.1.2 Delegation of Tasks

Delegation of tasks is somewhat inherent in the design of the pipeline, with each stage performing a distinct subset of the tasks in the framework. The OWEF class allows a definition of which tasks must be performed in order (word enumeration must preceed word scoring) and which may be performed in parallel (word clustering, word distribution). Additionally, the task of constructing the correct child object to fulfill each stage falls to the OWEF class. Apart from implementing the defined interfaces, a developer of a new child class must also add a check for their class to the appropriate stage creation section within the OWEF class. For example, the DataStructure object is initialized via the following code, making use of the special variable structure_type that is part of the OWEFArgs class (Listing 3.4):

```
if(list->count){
    if(list->structure_type.compare("rt") == 0)
            structure = new RadixTrie(list, input);
    else if(list->structure_type.compare("st") == 0)
            structure = new SuffixTree(list, input);
    else if(list->structure_type.compare("sa") == 0)
            structure = new SuffixArray(list, input);
}
```

Listing 3.4: Selection of DataStructure Object Type

A new developer wishing to add a new data structure, such as an STL Map approach, to the framework could add the following line below the SuffixArray definition to incorporate their new class into the framework (Listing 3.5):

```
    else if(list->structure_type.compare("ma") == 0)
            structure = new MapApproach(list, input);
```

Listing 3.5: Addition of a New DataStructure

This runtime modifiable approach allows multiple inherited child classes to coexist in the framework and be used on specific job types without the need for recompilation or code changes. A switch in the command line arguments easily selects the desired data structure or scoring model. This flexibility is important to the goal of user consistency in the framework, as a new word clustering method will require minimal changes to the user interface, whether graphical or command line, outside of the addition of a new option within the cluster_type switch variable already included in the OWEFArgs class.

3.4.2 The DataStructure Class

The following sub-sections of this chapter provide an overview of the general outline of the DataStructure class and also the requirements of all child classes of the DataStructure class and interfaces for use by other stages of the pipeline. Next, the internal sequence file class is discussed briefly to provide insight into the organization of the input file after being read into memory. Each currently implemented child of the DataStructure class is then discussed, with a focus on the RadixTrie class and, finally, the concept of a data structure iterator is discussed to allow systematic retrieval of all words stored in the database.

The DataStructure class provides a template through which a record of words may be constructed. Using the supplied input file, an object of the DataStructure class builds a custom database that may be queried by other objects within the framework. Since the DataStructure class is an abstract representation, a pure object of the DataStructure class cannot be created, rather a pointer is used and a child object's address is supplied to select the method by which the database of words will be stored in main memory. The specific storage mechanism for data is dependent on the algorithm chosen for the job. The algorithms currently implemented are discussed in subsequent sections.

3.4.2.1 Design and Primary Interfaces

There are several options that the user must specify for proper database creation, namely an input filename and the length of the words to be analyzed. The input filename may point to any accessible path, but the file must be in FASTA format [44]. This format allows multiple DNA sequences to be contained within a single file, while requiring a sequence identifier preceeded by a ”>” or ”;” character in between each sequence. While the word length parameter may be any integer, the resulting database can quickly become too large for a machine's main memory if this parameter is set too high. Due to the tool's

focus on DNA motif discovery, the word length is typically chosen to be between 6 and 20, although, during testing, lengths as long as 300 were successfully used for smaller datasets such as the *Arabidopsis thaliana* 3' UTR regions.

Additionally, a variety of options may be added to perform filtering of ambiguous nucleotides (n-filter), list nullomers (ancestral filter) or select the underlying data structure (structure type). Other less common options are available that allow the processing of a range of word lengths or set a minimum number of unique occurrences and minimum number of unique sequence occurrences required for a word to be present in the output of the framework. The n-filter and ancestral filter are both switches for which the framework requires no additional information, although the ancestral filter has a secondary switch (-e) to enumerate the nullomers, as, by default, this filter will occasionally output words in the form, ACCXXX, where the "X" character indicates that no words longer than ACC exist in the database with the prefix ACC. The enumeration switch will list all of the words that are not present in strictly DNA nucleotide characters, e.g. ACCAAA, ACCAAC, ACCAAG, etc. The processing of a range of inputs via the minimum length parameter is somewhat outdated, as it has been found to be unwieldy for users. Initially, this option was designed to allow multiple runs over a range of wordlengths to be processed in a single run, eliminating the need for rebuilding the database multiple times. In practice, however, users found it difficult to track their job parameters and seemed to prefer multiple runs for easier record keeping. This option is still available as a deprecated functionality and may be removed in future versions of the tool.

In certain cases, it may be interesting to look only at words that occur several times in the input, rather than viewing all words found. In this case, there are two parameters that may be used to limit the output of the framework. The first is the minimum occurrences option. This argument sets the minimum total number of times a word must occur in the input file. This option is often used when searching for motifs with high total occurrences

to eliminate the unimportant noise of words that occur only one or two times. Alternatively, the minimum sequences parameter counts only one occurrence per unique sequence in the input file, as there may be many sequences, and requires that a word be present in a specific number of sequences to appear in the output. An important use of this option is found in analysis promoter regions of known co-regulated genes when searching for an enhancer element that may be common to a large subset of those genes. It is important to note that, although these parameters are listed as data structure class parameters, the database will store, and report, all words in the database when queried. Filtering out the words that do not meet the minimum requirements is a task left to the later stages of the pipeline.

The data contains two private variables that are inherited by its children, ”OWEFArgs *list” is a pointer to the OWEFArgs object that is passed to each stage of the pipeline which holds all of the input parameters of the job at hand, allowing input-dependent processing to take place. The second private variable is ”SequenceFile *input”, which is a pointer to an abstract internal representation of the input file to ensure consistent processing across the stages and remove the duplication of input file storage.

The DataStructure class consists of a set of virtual interfaces which every child data structure must implement to operate within the framework. As a proof of concept, three DataStructure child classes have been implemented by members of the bioinformatics laboratory: RadixTrie, SuffixArray and SuffixTree. These structures are described in detail in the following sections. Table 3.1 lists the interfaces for the data structure class along with each function’s argument list, return value and a description of its purpose.

Table 3.1: The Standard Interfaces of the DataStructure Class

Function name	**Parameters**	**Return values**	**Purpose**
getCount	string motif	integer	Return the number of unique occurrences of a word in the database
getSeqs	string motif	integer	Return the number of unique sequence occurrences of a word in the database
getLocs	vector<int> &bit, string motif	void	Populates the bit vector with the sequences in which a word occurs

3.4.2.2 Iterators

An important set of classes related to the DataStructure class is the collection of iterators written to incrementally retrieve all information from the data structure used in a given job. These custom operators only require two functions, one is used to determine if more words are contained in the data structure and the other function is to incrementally retrieve the next word of a specific length (Table 3.2). Additionally, the DsIterator class provides a getCount function to determine the number of words available in the data structure. The iterator classes must maintain state information to properly return the next word in the data structure and to know when the last word has been returned to avoid accession errors. Each child of the DataStructure class must also have an iterator defined by the developer to properly integrate with the other stages of the framework. The RtIterator class, which defines the iterator used with the RadixTrie class, provides implementation level examples and is discussed in detail in the RadixTrie chapter.

Table 3.2: The Standard Interfaces of the DsIterator Class

Function name	**Parameters**	**Return values**	**Purpose**
getCount	void	integer	Return the number of words contained in the data structure
hasNext	void	boolean	Return true if there are more words in the data structure, false if not
next	void	string	Return the next lexicographic string from the data structure

3.4.3 The WordScoring Class

Once a database of words is obtained from the input file, the systematic scoring of every word in the database is performed by a child of the WordScoring class. This class uses statistical models to provide an approximation of how over- or under-represented a word is in the dataset. Currently, only Markov models [48] have been implemented as child classes to provide expected occurrence information, however, any other prediction model may be implemented as well, using the predefined interfaces discussed below. This section also discusses the general execution pattern expected for any WordScoring child and details the current implementations of the MarkovModel and RMESModel classes, which are the first two children integrated into the framework as child classes of the WordScoring class.

3.4.3.1 Design and Primary Interfaces

The WordScoring class is controlled through a set of user-specified options, similar to the DataStructure and other classes in the framework. The primary options select the appropriate child class to use for the current job and set the background model order for the Markov model. If another statistics model were to be implemented, a new option would need to be added for any additional parameters, such as the λ parameter in a Poisson distribution [48]. Optional switches include requesting additional information from the scoring model, such as providing a word's reverse complement and p-value as part of the scoring.

Since DNA is double stranded, biologists are often interested in an over-represented word, as well as its reverse complement. Figure 3.2 shows how the reverse complement of a DNA word is created. First, the word is written in reverse order, ACGT becomes TGCA. Then, each character of the word is replaced with its DNA binding partner, T for A, C for G, etc, so TGCA becomes ACGT. In this example, the reverse complement of the DNA

word is identical to the original DNA word. This special case is described as a palindrome in bioinformatics, similar to English language palindromes which are words that spell the same thing forward and backward. Palindromic sequences are of special interest since they are direction independent; they may bind the same protein whether the protein is scanning in the 5' to 3' direction or the 3' to 5' direction.

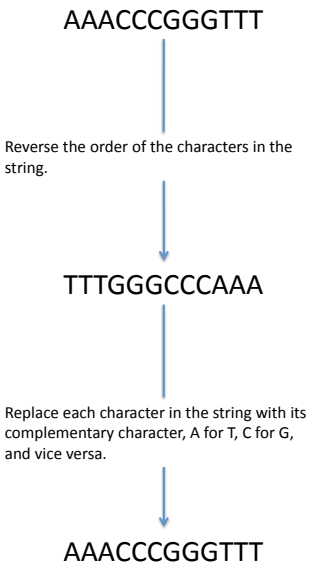

Figure 3.2: Creation of DNA Word Reverse Complements

The WordScoring parent class contains only two protected variables which are inherited by its children. The first is the OWEFArgs pointer that is passed to every stage of the pipeline for job parameters. The second is an array of three vectors, each containing pairs of words and their various statistics. This array is populated during scoring with the highest scored words of each of three scoring functions, $\frac{O}{E}$, $O * ln(\frac{O}{E})$ and $S * ln(\frac{S}{E_s})$, to provide the seeds to the clustering and other post-processing stages as necessary. The WordScoring class also defines two interfaces which may be used by other stages to

obtain scores for specific words (Table 3.3). The first interface, getSeeds, allows other stages to obtain a specific number of the highest ranking words in the dataset sorted by the scores previously mentioned. This function is implemented directly in the WordScoring class, and is not required as part of WordScoring child implementation. The function simply returns the appropriate vector from the top words array. The second interface, computeScores, takes a word and a pointer to a Scores object, which is an internal storage class for holding scores of various types, and calculates the appropriate scores for the word in question, then populates the variables inside the Scores object. This allows other stages to retrieve scores from the framework at any point.

Table 3.3: The Standard Interfaces of the WordScoring Class

Function name	**Parameters**	**Return values**	**Purpose**
computeScores	Scores *word, string motif, DataStructure *structure, int order	void	Populate the word pointer variables with the scores for the given motif
getSeeds	void	vector<pair<string, Scores>>	Return the vector containing the top words and their scores

3.4.4 The ClusterMethod Class

In addition to individual words, the identification of statistically relevant groups of words, or clusters, can aid biologists in the discovery of degenerate motifs [35, 36]. These motifs are groups of words with similar makeup and only minor changes at certain positions. The information in this section describes the process by which DNA words are generally clustered together and then discusses two specific clustering implementations currently present in the framework.

3.4.4.1 Design and Primary Interfaces

DNA binding proteins are important in the up- or down-regulation of genes outside the basal level inherent to the gene, as they may bind to specific enhancer sequences on the DNA to change expression levels. These proteins often have specific binding domains that are required for association with the DNA, but certain bases within that domain may be more flexible than others, leading to degenerate motifs. These motifs can be represented by clusters of similar words, with certain locations varying. This general concept of clustering has been well-studied and several clustering methods have been implemented to support user preference within the framework.

The ClusterMethod class contains the most execution of any of the three parent classes. In addition to the generic clustering of words, the ClusterMethod class and its children are responsible for the optional output of several useful pieces of information. The user may choose the number of clusters to be created, which scoring metric to use and the maximum distance a word may be from the cluster seed to still be contained in the cluster. Once the clusters are created, the class provides the option to output a position weight matrix (PWM) and regular expression representation of each cluster, as well as a graphical representation of the PWM and a statistical score for the cluster as a group, similar to the individual word scores provided by the WordScoring class. An additional

option allows dynamic selection of which score the so-called ”top words” are sorted by, $\frac{O}{E}$, $O * ln(\frac{O}{E})$ or $S * ln(\frac{S}{Es})$.

Unlike previous stages, no other areas of the framework directly request information from the ClusterMethod class. The interfaces listed in the ClusterMethod parent class are simply meant to abstract the basic functionality of the stage and ease the development of new methods based on other distance metrics. As a result, much of the code for the various clustering methods is implemented directly in the parent class, limiting the requirements for the child classes. As always in C++, however, any function in the parent class may be overridden by a child class and the context of the call will determine the proper version to use, as shown by the multiple definitions of the buildClusters function within the ClusterMethod, EditCluster and HammingCluster classes.

The ClusterMethod class defines a set of seven functions that each clustering method may make use of; only one is virtual and in need of explicit definition in child classes, and that is the buildClusters function (Table 3.4). The other functions are fully implemented within the parent class and may be called by any of the children as the command line options necessitate. The ClusterMethod class also provides a two-dimensional vector variable used for storing the clusters created.

Table 3.4: The Standard Interfaces of the ClusterMethod Class

Function name	**Parameters**	**Return values**	**Purpose**
buildClusters	DataStructure *structure, WordScoring *model	void	Populate the clusters variable and output the results
createLogos	DataStructure *structure, WordScoring *model	void	Create a visual logo representation of the provided cluster
scoreCluster	vector<string> cluster, DataStructure *structure, WordScoring *model	double	Calculate a score for the provided cluster
outputRegExp	ofstream &out_file, vector<vector<float> > pwm	void	Output the regular expression equivalent of the supplied Position Weight Matrix to the supplied file
outputPwm	ofstream &out_file, vector<vector<float> > pwm	void	Output the supplied Position Weight Matrix to the supplied file
computePwm	vector<string> cluster_data, DataStructure *structure	vector<vector<float> >	Output the supplied Position Weight Matrix to the supplied file
createMotifLogo	const vector<vector<float> > pwm, const string &filename	void	Output the supplied Position Weight Matrix to the supplied file

3.4.5 Additional Functionality

Following the completion of the primary stages of the pipeline, it is often desirable to perfrom several informative post-processing steps. These small command line scripts are visualizations of the results, providing at-a-glance interpretations of the information useful for reporting and publications. The WordDistribution class focuses on a few of the top scored words and visualizes each word's individual locations across the various input sequences, while the scatter plots show the ranges of $O * ln(\frac{O}{E})$, $S * ln(\frac{S}{E_s})$ and p-value scores of all the words reported, allowing identification of outliers and a measure of additional confidence to the results. Further stages have been added as examples of extension to the framework and, although these classes are not fully functional, they provide a useful example for new developers to use when completing the functionality or, more importantly, adding new extensions to the framework. These classes include the SequenceClustering class, the FunctionalLookup class, ConservationAnalysis class and the ModuleDiscovery class. Finally, the NewTemplate class is described, which provides a basal level description of how to extend the framework through simple descriptions and naming convention examples.

3.4.5.1 The WordDistribution Class

The WordDistribution class is useful for showing the locations of a single word at various positions within the unique sequences in a dataset. For example, a word which always occurs within ten base pairs of the start of a sequence would stand out in the plot due to the accumulation of datapoints related to these occurrences. The output can be formatted to show absolute position or normalized position, the former is useful for sequences of similar length, while the latter is often used when sequence length varies greatly across sequences to produce an average position for the words. Typically, word distributions are only created for a select few top-ranked words in the output, as full

analysis would be prohibitively time consuming. Once interesting words are identified, the analysis of their distribution throughout the sequences can provide additional insight into their function and value as research targets.

Word distributions are an optional stage of the pipeline, with a user flag to cause their creation, along with a variable selecting the number of top words for which the system will produce distribution information, and an option to normalize the distributions to account for variable sequence length.

3.4.5.2 Creation of Scatter Plots

Another useful output of the OWEF framework is a set of scatter plots that show the relative scores of all words within the input dataset. These plots can be used to identify outliers from the dataset for further inspection, and are useful in publications as visual representations of the text. The scatter plots are created via an additional command line flag, and are implemented in the system as external calls from the C++ application to shell scripts used to control gnuplot [1]. Four scatter plot scripts are included with the framework, one to plot the $O * ln(\frac{O}{E})$ scores of the words in the input, one to plot the $S * ln(\frac{S}{E_s})$ scores, a third to plot the p-value of the words and, finally, a plot for the expected counts of any words that do not occur in the input. These expected scores can be computed using the sub-word frequencies to predict the number of times the missing word should have occurred statistically.

3.4.5.3 The Shell Classes

The three shell classes currently contained in the OWEF framework are SequenceClustering, FunctionalLookup and ConservationAnalysis. These classes implement only a very basic constructor and destructor, and a call to a function that can be used by developers to complete the functionality. These functions are called clusterSequences, lookup and findConservation, respectively. The SequenceClustering

class will include logic to group together sets of sequences from the input dataset that have many, or specific, words in common. This is useful information for users looking for co-regulated sequences within a large dataset, which can be important to proper regulation of stress response genes as shown in [34]. The FunctionalLookup and ConservationAnalysis classes will both be used to poll existing databases for additional information about words found in the input. FunctionalLookup will perform database queries against the Gene Ontology database [5], while ConservationAnalysis will use databases such as Jaspar [52] and Transfac [69] to add contextually relevant information. The ModuleDiscovery class has been added to the WordSeeker tool as part of research work on sets of regulatory elements. This stage looks into word modules [34]; groups of two or more words, that may be related to gene regulation, providing additional information about how these groups may aid gene expression level control.

3.4.5.4 The NewTemplate Class

This class is never actually compiled into the framework, but, instead, provides insight to new developers into how one goes about adding new stages to the toolkit. Heavily commented, the NewTemplate class provides a declaration of the class, including a basic constructor, an initialized constructor that passes objects typically required by stages, and a destructor. In addition to this pre-defined code, the NewTemplate class is heavily commented to direct the user on how to approach extension of the framework and give insight into the naming and coding conventions used in the framework and, hopefully, to help avoid confusion.

4 Multi-Level Parallelization

The framework includes code for both single-node and distibuted versions, selected at compile time by a flag. To compile the single-node codebase, the user simply uses a typical ”make; make install” command to build the output files and finally the executable. This build requires a standard g++ compiler, 4.1.2 or higher, and, by default, OpenMP [13] compatibility, although this can be changed by modifying the CPPFLAGS within the Makefile. To build the distributed system, the user must add ”PLAT=multi” to the make commands, which causes the system to use the distributed code definitions, which remove the dependency on OpenMP, but add the requirement of an MPICH [23] distributed environment. A debug version of each system is also possible through the addition of ”DEBUG=yes” to the make system commands, which flags the executable for profiling and GDB debugging options. The available make commands are (Listing 4.1):

```
Sequential, no debugging:
    make; make install
Sequential, debugging on:
    make DEBUG=yes; make DEBUG=yes install
Distributed, no debugging:
    make PLAT=multi; make PLAT=multi install
Distribute, debugging on:
    make DEBUG=yes PLAT=multi; make DEBUG=yes PLAT=multi install
```

Listing 4.1: Make Command Options

4.1 Parallelization Strategies

Two primary models have emerged for parallel programming, shared memory [13] and distributed memory [10]. Shared memory applications rely on multiple processors, either several discrete physical processors or a single processor with multiple logical cores

or, recently, multiple discrete processors with several logical cores on each chip. Applications using shared memory run on a single machine, and each processor core has access to the available main memory on the machine, hence the shared memory nomenclature. Distributed memory applications, on the other hand, operate on a networked system of computers, with each machine performing a sub-task of the total solution, using interprocess communication libraries such as MPI (Message Passing Interface) [23] or RPC (Remote Procedure Call) [6] commands to obtain necessary information from other nodes in the system. There are many methods to use to split the work, including task decomposition, data decomposition and data-flow decomposition; the OWEF framework utilitzes task decomposition techniques and minor data-flow enhancements, which will be discussed in the following sections.

4.1.1 Shared Memory Approach

Shared memory programming has the advantage of being traditionally easier to implement, as code libraries have been developed for a kind of ”automatic” parallelization. One example of such a library is the Open MP library used in the OWEF framework. This set of commands can be used to perform simple parallelization of well-defined for-loops in code, utilizing the system's multiple processors to reduce the time required to run the loop. The Open MP specifications [13] provide a list of macros and function calls associated with the Open MP system, only a few of which have been used to parallelize the OWEF codebase. A loop that is parallelized with Open MP must have a strict, unchanging exit condition; at no point in the loop should the code perform changes to the loop counter variable except for the standard increment at the end of each loop iteration. Additionally, certain lines of code within a loop may be marked as critical, meaning that only one thread may enter that section of commands at a time, which

provides a simple locking scheme. More complex locks are available, but were not used as part of this project.

Listings 4.2 and 4.3 show a typical for-loop before and after parallelization with Open MP. Several changes are necessary to get the for-loop ready for parallelization. For the Open MP code to know about the loop counter variables, they must be declared prior to the loop declaration. In addition, variables may be provided as shared or private. The shared variables in this example are the outer loop counter, necessary to allow the loop to be split into threaded tasks, and the calculations vector. This allows all the threads to modify the same data structure without needing to add any extra communication at the end of the loop. Each thread receives its own copy of the private variables, which are in this case listed as firstprivate, meaning that the variable copies are initialized to the value of the source variable prior to entering the for-loop. The value of the source variable is undefined after the loop. Alternatively, a private variable may be listed as lastprivate, indicating that the last thread to exit the loop sets the value of the source variable so it may be used after the for-loop. In this case, the value of the private variables is undefined on entry to the for-loop (the variables must be initialized within the loop).

```
Initial For-loop:
//local variables
vector<double> calculations;
//for loop before parallelization
for(int i=0; i<exitCondition; i++) {
   for(int j=0; j<exitCondition2; j++) {
      double temp = 0;
      temp += (double)i*(double)j/2.0;
   }
   calculations[i] = temp;
}
```

Listing 4.2: Example of General Purpose For-loop

```
After adding Open MP macros, the loop becomes:
//new local variables
vector<double> calculations;
int i, j = 0;
#pragma omp parallel for shared(i, calculations) firstprivate(j)
for(i=0; i<exitCondition; i++) {
   for(j=0; j<exitCondition2; j++) {
      double temp = 0;
      temp += (double)i*(double)j/2.0;
   }
   #pragma omp critical {
   calculations[i] = temp;
   }
}
```

Listing 4.3: Example of For-loop with Parallelization

4.1.2 Distributed Memory Approach

Due to memory restrictions on single nodes, the distribution of tasks has been the primary focus of the parallelization of the OWEF framework. The Bio-cluster at Ohio University has five nodes, each with eight processor cores and 32 GB of main memory, which is considerably more than most desktop computers. Even these powerful nodes, however, quickly succumb to the intense demands of processing entire genomes, necessitating a distributed approach. Using the MPI libraries available on most HPC (High Performance Computing) systems, the task of distributing the work focused on two key concepts: minimize the required network communication and minimize data redundancy. Both of these goals can be realized by distributing through a prefix-based task assignment. The breakup of the job into prefix-based tasks is discussed in section 4.1.2.1

4.1.2.1 Prefix-based distribution

The DNA alphabet is a restricted subset of the English alphabet, containing the characters A, C, G, T and, occassionally, N. The IUPAC characters [12] do contain specifications for several additional characters; however, occurrences of these characters are internally mapped to the ambiguous nucleotide, N, in the framework. This limited alphabet provides an ideal platform for the distribution of word enumeration tasks. In the simplest case, five nodes are needed, one for each nucleotide character. Each node is tasked with processing only those words in the dataset that begin with its assigned character, or prefix. For example, Node 0 would process words beginning with the ”A” prefix, Node 1 the ”C” prefix and so on. This process creates five independent nodes, none of which have any words in their databases that also occur on other nodes, completely eliminating data redundancy in the nodes’ main memory. Because a target word only occurs on a single node, this distribution also reduces network overhead by allowing direct communication between nodes, eliminating expensive broadcast messages when

information is needed about a word. Figure 4.1 shows how tasks are distributed among the nodes during a distributed memory run.

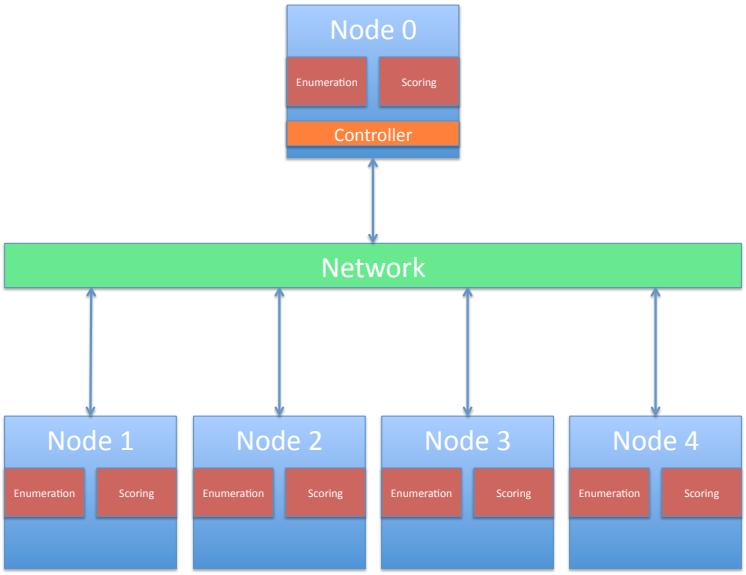

Figure 4.1: Prefix distribution in the OWEF Framework

Because the data is independently distributed among the nodes, an input dataset with relatively even percent makeup of A, C, G, T and N characters would cut the memory footprint on a given node down to 20% of the memory required to process that job on a single node. The total main memory used remains the same, it is just evenly distributed among the various nodes. In practice, the occurrence of the N character is rare, and typically the memory footprint is cut to approximately 25% of the original footprint, allowing the distributed system to process much longer word lengths than the shared memory version. Depending on the data structure used for the job, the maximum possible word length can be important. For large datasets, the SuffixTree class may not be capable of building the entire structure, but, if it can, there is no limit on maximum word length,

up to the length of the longest sequence in the input. In this case, lowering the memory footprint required for each node gives the SuffixTree class a better chance of fully enumerating the data. For the RadixTrie, on the other hand, the size of the structure increases dramatically as word length increases, and lessening the memory footprint will allow the RadixTrie class to process longer word lengths than previously possible. This example shows that not all data structures grow in size equally. Suffix trees are dependent on the size of the input file, not the word length requested, while radix tries are dependent on the word length. If the radix trie can fit in main memory for a given word length with all possible words present, doubling the size of the input file will have no effect at all on the ability to process the file.

Each prefix requires two tasks, one for the enumeration of words, which builds the data structure and serves information about the content to all other tasks in the distributed system, and one to perform all of the other stages of the pipeline, such as word scoring and clustering. MPI distribution typically uses one task per node, but, in this case, the framework utilizes two tasks, or more, per node. Currently, the number of nodes and tasks-per-node are fixed at five and two, respectively, but as deeper prefix options are introduced, these will become dynamic concepts, allowing the system to better scale to its environment.

It is easy to see how this prefix concept could be extended to longer prefixes and with additional nodes involved in computation. A prefix of length two would require $4^2 + 1$, or 17 nodes, and the general equation for a prefix length of n is provided by the equation $4^n + 1$. More specifically, prefix depth two would require thirty-four tasks (two per prefix) not nodes, but, as the selection of prefix depth is related to lightening the strain on a node's main memory, using less nodes would only add network traffic and slow down the system, without increasing the capabilites. The addition of these deeper prefixes will introduce some data redundancy at word lengths less than the prefix length, but relative to

the size of the data structure as a whole, these few redundant words are not a concern from a memory standpoint. The extension of the codebase to support an arbitrary prefix is in development, but is not currently implemented in the latest release of the framework. The communication of stages between nodes is discussed in more detail in the following sections, as each stage is examined and its specific distribution approach is described.

4.1.3 Combined Approach

The combination of shared memory and distributed memory parallelization approaches is often referred to as multi-level parallelism, where tasks are distributed to a system of multicore machines and each machine then implements some amount of parallelization within its task. This technique requires significantly more planning and more elaborate locking schemes to avoid deadlocks and other obstacles. Currently, planning is underway to add this functionality into the OWEF framework, but no significant progress has been made. The concept is simple, as both shared memory and distributed memory versions of the code are in place. In general, the framework has been distributed effectively, and the minor shared memory enhancements have improved runtimes, however, inconsistencies were identified when attempting to combine the two approaches, and this enhancement was delayed in favor of other, higher priority work.

4.2 Parallelization of the OWEF Class

Once a hardware decision is made and the code is compiled, the OWEF class contains two unique constructors, one for each version of the code. Most stages of the pipeline contain one constructor with a mix of instructions, but the complexity of task delegation present in the logic of the OWEF class made separate constructors useful and the code easier to read. The following subsections provide detailed descriptions of the sequential and distributed constructors.

4.2.1 The Shared Memory Constructor

The shared memory constructor is initialized with a pointer to an OWEFArgs object which is first copied to an internal pointer to an object of the same type. This allows all functions of the OWEF class to access the command line arguments without passing the pointer from function to function. Next, a pointer to a SequenceFile object is created and initialized with the filename provided in the OWEFArgs class. The SequenceFile class is an internal representation of the file using string pointers that can provide detailed information about the set of sequences contained in the input, such as the length of each sequence, the FASTA header for each sequence and the strings contained in each sequence. The SequenceFile class also contains several filters such as the ancestral filter, which removes sections of sequences previously marked as ancestral repeats [59] and the ”N” filter, which removes all ambiguous nucleotides from the input. The inclusion of these filters in the SequenceFile class eliminates the need for redundant programming in the DataStructure child classes, and provides consistency throughout the framework. The OWEF class uses this object to build background information for the statistical models implemented in the WordScoring child classes by populating the total_input_length and background_seqs variables for later calculations. Following the initialization of these variables, the DataStructure, WordScoring, WordClustering and other objects are created as previously described. Finally, control is returned to the main function for cleanup and organization.

4.2.2 The Distributed Constructor

In addition to the setup steps for the OWEFArgs object that are identical to the shared memory constructor, the distributed constructor also contains logic to allow distribution of tasks to multiple computational nodes. The framework provides distributed computation utilizing five multicore nodes and divides the work among the nodes using the

aforementioned prefix-based approach. The focus on DNA words allows each of the five nodes to process only the words in the input file with the prefix it is assigned: A, C, G, T or N (Figure 4.2). This assignment not only reduces the memory load on each node, but the prefix approach eliminates redundancy among the nodes, as each word only occurs on one node, lightening the communication overhead typically incurred with a distributed approach. For a given node, two MPI tasks are initialized. Task 0 performs the database construction while task 1 waits on notification that the input has been fully processed and the database is constructed. This is necessary, as the WordScoring class iteratively requests each word in the database from the DataStructure class, requiring a complete database prior to the start of the scoring stage. Once the database is built, task 0 posts an MPI receive instruction and waits on information requests from any task in the system in a server-client situation. Any node may receive a request for information from any other node, and a number of different requests may be made (Table 4.1).

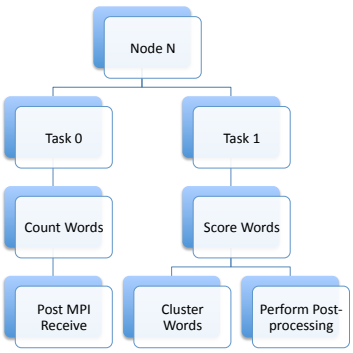

Figure 4.2: Task Distribution in the OWEF Class

These requests provide a consistent interface between nodes that allows full access to the information contained in the database. Each node's task 0 keeps a local iterator to the DataStructure class object for each other task in the MPI process; currently there are ten tasks in the distributed setup, so each node has an array of ten iterators keeping track of its state. This allows each task to process at its own pace, independent of the speed of other tasks, eliminating the need for costly locking and synchronization. The MPI_NEXT_WORD_TAG, MPI_COUNT_TAG, MPI_SEQS_TAG, MPI_REGEX_TAG and MPI_LOCS_TAG are distributed versions of the standard interfaces to the DataStructure class, and the OWEF simply receives the request, calls the standard interface function and sends the returned information back to the requesting task. The MPI_DONE_TAG and MPI_RESET_TAG are control tags to provide proper operation of the OWEF class, and the MPI_NUMWORDS_TAG allows the scoring stages to know how many words are in a specific database to provide a loop exit condition. In the distributed system, no scoring task ever requests information directly from a database, even if the database being polled is on the same computational node. Requests are always made to the OWEF class on the appropriate node, which passes the request on to the database. This adds a level of communication complexity to the system, but since it is possible to calculate the node which contains the information, broadcast messages are not necessary and the built-in optimizations in MPI modify the intranode communication by recognizing identical source and destination ID's and using optimized communication functions. Listing 4.4 shows the general control flow of the OWEF distributed constructor, and full descriptions of how these interfaces are used in the individual stages are described in the following sections.

```
Set up background statistics
for(i=0:number of nodes) {    //5
   for(j=0:number of tasks per node) {     //2
      Determine prefix
      if(counting task) {
         Create DataStructure child object
         Wait on all databases to be created
         Create the appropriate iterator for the DataStructure in use
         while(!done) {
            Receive MPI request
            if(MPI_NUMWORDS_TAG) {
               Send number of words in database via MPI
            } else if(MPI_NEXT_WORD_TAG) {
               Request next word from local iterator
               Send next word from iterator via MPI
            } else if(MPI_COUNT_TAG) {
               Request count of provided word from local database
               Send count of provided word via MPI
            } else if(MPI_SEQS_TAG) {
               Request unique sequence count from local database
               Send unique sequence count via MPI
            } else if(MPI_REGEX_TAG) {
               Request regex matches from local database
               Send number of matches via MPI
               for(k=0:number of matches) {
                  Send match k via MPI
               }
            } else if(MPI_LOCS_TAG) {
               Request location vector population from local database
               Send vector size via MPI
```

```
                for(k=0:size of vector) {
                    Send index k via MPI
                }
            }
        }
    }
    else {     //do scoring
        if(scoring)
            Create WordScoring child object
        if(clustering)
            Create WordClustering child object
        if(computing word distributions)
            Create WordDistribution object
    }
  }
}
```

Listing 4.4: Control Flow of the Distributed OWEF Constructor

Table 4.1: MPI Tags for Distributed Interprocess Communication

Tag	**Purpose**
MPI_DONE_TAG	Allow database to exit normally
MPI_NUMWORDS_TAG	Request the number of words contained in the database
MPI_NEXT_WORD_TAG	Request the next word iteratively from the database
MPI_COUNT_TAG	Request the count of unique occurences of a word from the database
MPI_SEQS_TAG	Request the count of unique sequence occurrences of a word from the database
MPI_REGEX_TAG	Request all words from the database that match a given regular expression
MPI_LOCS_TAG	Increment a bit vector to indicate which sequences a word occurs in
MPI_RESET_TAG	Reset the OWEF class's iterator so the next request retrieves the first word

4.3 Parallelization of the DataStructure Child Classes

DataStructure children are a special case in the OWEF framework, as they do not need to have any explicit knowledge of whether the system is running in shared or distributed memory, they simply need to be able to process only a subset of the input based on a supplied prefix when directed. The OWEF class and child classes for other stages control the distribued logic, always requesting information from the appropriate node, guaranteeing a one hundred percent hit rate, assuming the word being queried occurs in the original input. In the RadixTrie class, a simple check against the prefix during sequence processing allows the loop to operate for both versions of the framework (Listing 4.5). As the #ifdef statement shows, this section of code is only included by the compiler when the make system is directed to build the distributed version of the framework and, while the logic of the rest of the input processing loop remains unchanged, this allows the radix trie to be built for only a specific prefix on a given node. The other DataStructure child classes perform prefix filtering in a similar manner.

```
#ifdef KKURZ_MPI
if(toupper((*segs[x])[i]) != list->pref)
        continue;
else
{
#endif
```

Listing 4.5: Checking Against the Prefix in the RadixTrie Class

In the early stages of incorporating a new data structure into the framework, it is possible to run the distributed system with every node containing the full data structure, ignoring the prefix provided. This is useful for development, but obviously highly inefficient. Also, output may be extremely redundant, as the iterators for each node will

return all words, rather than just words beginning with the assigned prefix. This is useful in preliminary testing, but all data structures must be able to build a prefix-based structure containing a partitioned subset of the words found in the input for proper execution through the framework logic.

4.4 Parallelization of Later Stages

Early revisions of the OWEF framework laid the bulk of network communication on the DataStructure child classes, but this proved to be inefficient and difficult to manage. The decision was made to include MPI communication in the other stages, as this allowed a database task to build the data structure and then serve requests from the other tasks in the system. In general, any calls to the standard interfaces included in the DataStructure class definition must be placed within a macro to include version specific code at compile-time (Listing 4.6).

```
//retrieving the number of words from the database
#ifdef KKURZ_MPI
int stat;
rc = MPI_Send(&buf, 0, MPI_CHAR, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD, &status);
stat = i+list->minlength;
rc = MPI_Send(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&rec, 1, MPI_INT, list->hosts.host_array[branch_array[list
    ->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD, &status);
numWords = rec;
#else
numWords = list->num_words[i + list->minlength -1];
#endif
```

```
//retrieving the next word from the database
#ifdef KKURZ_MPI
rc = MPI_Send(&buf, 0, MPI_CHAR, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NEXT_WORD_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&buf, i+list->minlength, MPI_CHAR, list->hosts.host_array[
    branch_array[list->pref - 'A']][0], MPI_NEXT_WORD_TAG,
    MPI_COMM_WORLD, &status);
next = "";
next += buf;
#else
if(it->hasNext())
    next = it->next();
#endif

//getting the count of a word from the database
#ifdef KKURZ_MPI
rc = MPI_Send(&motif[0], motif.length(), MPI_CHAR, list->hosts.
    host_array[branch_array[motif[0] - 'A']][0], MPI_COUNT_TAG,
    MPI_COMM_WORLD);
rc = MPI_Recv(&rec, 1, MPI_INT, list->hosts.host_array[branch_array[
    motif[0] - 'A']][0], MPI_COUNT_TAG, MPI_COMM_WORLD, &status);
word->count = rec;
#else
word->count = structure->getCount( motif );
#endif

//getting the sequence count of a word from the database
#ifdef KKURZ_MPI
```

```
rc = MPI_Send(&motif[0], motif.length(), MPI_CHAR, list->hosts.
    host_array[branch_array[motif[0] - 'A']][0], MPI_SEQS_TAG,
    MPI_COMM_WORLD);
rc = MPI_Recv(&rec, 1, MPI_INT, list->hosts.host_array[branch_array[
    motif[0] - 'A']][0], MPI_SEQS_TAG, MPI_COMM_WORLD, &status);
word->seqs = rec;
#else
word->seqs = structure->getSeqs( motif );
#endif

//getting the specific sequences a word occurs in from the database
#ifdef KKURZ_MPI
rc = MPI_Send(&clusters[i][j][0], clusters[i][j].length(), MPI_CHAR,
    list->hosts.host_array[branch_array[clusters[i][j][0]-'A']][0],
    MPI_LOCS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    clusters[i][j][0]-'A']][0], MPI_LOCS_TAG, MPI_COMM_WORLD, &status);
for(int y=0; y<stat; y++) {
    rc = MPI_Recv(&rec, 1, MPI_INT, list->hosts.host_array[branch_array[
        clusters[i][j][0]-'A']][0], MPI_LOCS_TAG, MPI_COMM_WORLD, &status
        );
           bit[y] = rec;
}
#else
    structure->getLocs(bit, clusters[i][j]);
#endif
```

Listing 4.6: Generalizing DataStructure Calls

The distributed code for getting the number of words is more intricate than one would expect due to limitations of the MPI_Recv function. This function requires an explicit variable type to be specified. Requesting the number of words from the database

requires the length of the words in question, as the database contains words of length one up to the maximum word length requested by the user. Since every other request from the database is done with a string, the MPI_NUM_WORDS_TAG section includes a handshake where the database first receives the notification that the number of words is being requested, then sends back an integer, set to one, indicating that it has posted the receive command allowing the requesting process to send the word length desired. Once the word length has been sent, the database returns the integer for the number of unique words contained in the database.

Getting the specific sequences a word occurs in from the database also requires a small amount of extra code. First, a request is sent to the database for the sequence locations of a word and a receive is posted. The integer received is the size of the vector that is about to be sent by the database. The for-loop then receives the integer value for each position in the vector and sets the variables correctly.

4.4.1 Parallel Code in WordScoring Child Class

The children of the WordScoring class must then contain several macros for code definition based on whether the code is compiled for shared or distributed memory runs. While the inclusion of both versions within each function makes the code more difficult to read, it significantly lowers code redundancy and retains a single codebase for the entire framework. Listing 4.7 shows the general task of word scoring, and Listing 4.8 shows the code modified with checks for distributed memory code compilation.

```
int numWords = list->num_words[i + list->minlength -1];
iterator *it = new iterator;
for(int i=0; i<numWords; i++) {
    string next;
    if(it->hasNext())
        next = it->next();
    Scores *word = new Scores;
    computeScores(word, next, structure, list->order);
    output(word, next);
}
```

Listing 4.7: General Logic for Word Scoring

```
int numWords = 0;
#ifdef KKURZ_MPI
int stat;
rc = MPI_Send(&buf, 0, MPI_CHAR, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD, &status);
stat = i+list->minlength;
rc = MPI_Send(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&rec, 1, MPI_INT, list->hosts.host_array[branch_array[list
    ->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD, &status);
numWords = rec;
#else
numWords = list->num_words[i + list->minlength -1];
iterator *it = new iterator;
```

```
#endif


for(int i=0; i<numWords; i++) {
    string next;
#ifdef KKURZ_MPI
    rc = MPI_Send(&buf, 0, MPI_CHAR, list->hosts.host_array[branch_array[
        list->pref - 'A']][0], MPI_NEXT_WORD_TAG, MPI_COMM_WORLD);
    rc = MPI_Recv(&buf, i+list->minlength, MPI_CHAR, list->hosts.
        host_array[branch_array[list->pref - 'A']][0], MPI_NEXT_WORD_TAG,
         MPI_COMM_WORLD, &status);
    next = "";
    next += buf;
#else
    if(it->hasNext())
       next = it->next();
#endif
    Scores *word = new Scores;
    computeScores(word, next, structure, list->order);
    output(word, next);
}
```

Listing 4.8: Distribution-aware Word Scoring

As shown in Listing 4.8, only minor changes are needed to the main loop that processes all of the words in the database. The first change is to request the number of words from the distributed database, rather than asking the local database directly, and the second is to iteratively request the next word from the distributed system, again replacing the local call with an MPI request. In a similar fashion, other functions in the WordScoring children that need information about a word from the database need to have two versions of the request, one for a local, shared memory approach, and one for a distributed MPI approach.

4.4.2 Parallel code in ClusterMethod Child Classes

Similar to the WordScoring children, child classes of the ClusterMethod class must implement two versions of certain instructions to account for the duality of the codebase as it handles both shared memory and distributed memory methodologies in a single file. To reduce the time constraints of creating all possible word clusters, a subset of the words contained in the database is used as the set of seed words for clusters. This set of words can be obtained from the local WordScoring model pointer provided to the clustering object. The set of all words is then compared to each of the seed words to find the subset of words that are within the specified distance of a given seed word. Again, any DataStructure class interfaces used must be implemented for both the shared and distributed memory versions (Listing 4.9).

```
int numWords = 0;
#ifdef KKURZ_MPI
int stat;
rc = MPI_Send(&buf, 0, MPI_CHAR, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD, &status);
stat = i+list->minlength;
rc = MPI_Send(&stat, 1, MPI_INT, list->hosts.host_array[branch_array[
    list->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD);
rc = MPI_Recv(&rec, 1, MPI_INT, list->hosts.host_array[branch_array[list
    ->pref - 'A']][0], MPI_NUMWORDS_TAG, MPI_COMM_WORLD, &status);
numWords = rec;
#else
numWords = list->num_words[i + list->minlength -1];
iterator *it = new iterator;
```

```
#endif

vector<pair<string , Scores> > my_seeds = model->getSeeds();
for( int i = 0; i < static_cast<int> ( my_seeds.size() ); i++ ) {
   vector<string> temp;
         temp.push_back( my_seeds[ i ].first );
         clusters.push_back( temp );
}
vector<vector<string> > t(clusters);

for( j = 0; j < static_cast<int> ( t.size() ); j++ ) {
   for( i = 0; i < numWords; i++ ) {
#ifdef KKURZ_MPI
      rc = MPI_Send(&buf, 0, MPI_CHAR, list->hosts.host_array[x][0],
         MPI_NEXT_WORD_TAG, MPI_COMM_WORLD);
      rc = MPI_Recv(&buf, i+list->minlength, MPI_CHAR, list->hosts.
         host_array[x][0], MPI_NEXT_WORD_TAG, MPI_COMM_WORLD, &status);
      next = "";
      next += buf;
#else
      if( it->hasNext() )
         next = it->next();
#endif
      if( hammingDist( clusters[ j ][ 0 ], next ) <= list->distance )
         clusters[ j ].push_back( next );
   }
}
```

Listing 4.9: Distributed-aware Word Clustering

5 A Detailed View of The RadixTrie Class

Originally, the WordSeeker tool was built using the Teiresias algorithm [49] as a backend database for words. This algorithm provides a list of maximal patterns as its output, making it difficult to enumerate words of a specific length. Rather, Teiresias groups words together if the one word is a substring of the other word, and the longer word appears exactly as many times as the shorter word. For example, if the word ACGT occurs five times in an input dataset, and the word ACGTGGT also occurs five times, ACGT will not be reported by Teiresias because the longer word, ACGTGGT, fully encompasses all occurrences of the shorter word. When the decision was made to fully enumerate the input space for a user-specified wordlength, the Teiresias algorithm was deemed to be overly complex. Additionally, Teiresias stores its words in an array, leaving a large memory footprint for large input sets. The search for a simpler data structure capable of reporting all unique words found in the dataset in a space-efficient manner brought several possible algorithms to the table, one of which was the radix trie [16], due to its ability to encode words in a tree-like structure, provide fast access times and the existence of a simple partitioning for parallel processing. The RadixTrie class was built as a DataStructure child class to implement the algorithms for construction and querying of a radix trie structure using the DNA alphabet.

5.1 General Concepts

The radix trie is conceptually similar to a suffix tree, or any general tree structure. The structure is initialized with a root node, and each time a word is added to the tree, nodes are added to hold each character of the word (Figure 5.1). This allows the radix trie to store not only the information about words of the selected length, but also information about all shorter words, without requiring any additional space for the substrings.

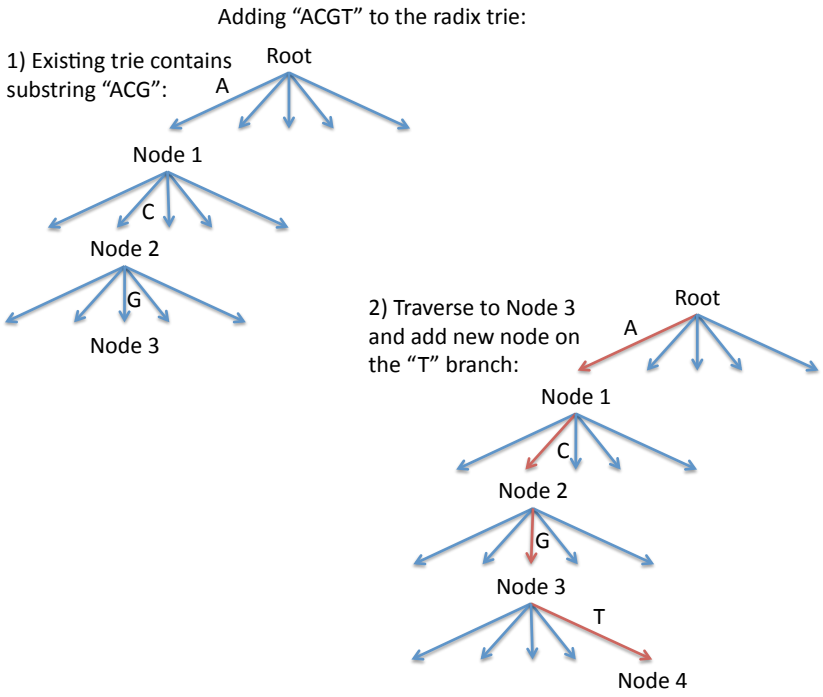

Figure 5.1: Building the Radix Trie

The radix trie structure benefits greatly from limited alphabet size, as each node must store pointers to all possible characters for the next position in the string, making a node in a radix trie with an alphabet of size five (DNA, including N’s) much smaller than a node in a radix trie with alphabet size twenty-six (English text). A radix trie can retrieve the information about a word in $O(n)$ time, where n is the word length. The structure of the radix trie does not explicitly store any strings, the strings are built while traversing the trie and each step from one node to another represents adding a character to the string (Figure 5.2). Once the traversal builds the requested string, the data associated with the query string is obtained from the terminal node, and may be returned to the requesting function. This implicit storage of strings aids the space efficiency of the structure, although the need

to store branch pointers still necessitates some optimizations, discussed in section 5.4. The radix trie, in general, has a worst-case size requirement of $O(N^2)$ space, where N is the length of the input. This space requirement describes a trie where every possible word of length n was present in the input dataset and n is equal to N. In practice, the trie is never this large, and a better approximation for small n is $O(\sum_{i=1}^{n}(4^n) * sizeof(RadixTrieNode))$. This definition is born out by Table 5.1, which shows the reliance of the radix trie on how many words of the requested length occur in the input, rather than input size itself, as a fully populated trie with a small input set will not be any larger than a fully populated trie with, for example, the entire human genome.

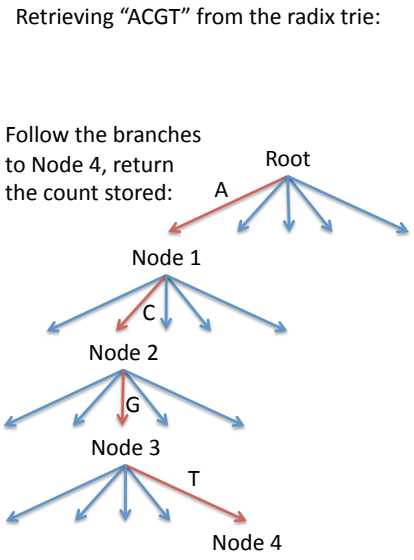

Figure 5.2: Retrieving a string from the Radix Trie

Table 5.1: Space Requirement Examples for the Radix Trie

File	**Size** (MB)	**Word Length**	**Number of Words**	**Trie Size** (MB)
A. thaliana Core Promoters	3.8	8	65151	2.962
A. thaliana Intron Regions	25	8	65470	2.969
A. thaliana Full Genome	116	8	65536	2.971
H. sapiens Full Genome	3095	8	65536	2.971

Obviously, as the trie becomes less densely populated (which is typical as word length increases), even this size estimate is overly conservative, allowing the RadixTrie class to process very large input for long word lengths. The implementation of parallel word enumeration, discussed in Chapter 4, divides the space requirements for the radix trie among several nodes, making it possible to operate on even longer word lengths.

The nodes of the RadixTrie class contain several variables used for reporting and status checks. A node stores the number of total occurrences, the number of unique sequences and the last sequence where the associated word was found. This last variable allows the building phase to know when to increment the unique sequences counter, as a word may occur several times in each sequence. In addition to these data variables, each node contains a pointer to an array of RadixTrieNode pointers, allowing the node to extend the trie when needed.

5.2 Building the Structure

Providing the information required by word enumeration tools built on the OWEF framework requires that the data structure implementation be able to systematically return all words of a specific length that occur in the input, as well as provide access to any statistical information available about the words contained in the data structure, including words shorter than the user-specified word length. The RadixTrie class provides this

functionality by building a trie containing all words from length one up to the word length, storing unique occurrences, unique sequence occurrences and last viewed sequence for each word. Using this information, obtained through the standard interfaces described in the DataStructure class, subsequent scoring models can compute valuable statistics about each word found in the input.

To build the structure, the initialized constructor is called, which performs basic setup of the object and then calls the countWords function to build the data structure. Four main functions are used to build the database: countWords, incCount, trieGet and trieAdd, allowing the initial creation and further extension of the trie using the supplied SequenceFile object, which maps the input sequence into main memory. Table 5.2 shows these three functions' input and output parameters, and the following subsections describes each function in detail.

Table 5.2: Functions for Building the Radix Trie

Function name	**Parameters**	**Return values**	**Purpose**
RadixTrie	OWEFArgs *from_input, SequenceFile *in	N/A	Initialized constructor tasked with building the structure
incCount	char *motif, int length	int	Increment the count of the word and, if necessary, the sequence count
countWords	SequenceFile *input	void	Count all the words contained in the input file and store in the radix trie
trieGet	RadixTrieNode *&node, char *s, int length, int level	int	Search the trie for a word, if found increment count and return previous count. If not, return (-1)
trieAdd	RadixTrieNode *&node, char *s, int length, int level	void	Adds a word to the trie, adding any intermediate nodes needed to reach the appropriate length.

5.2.1 The RadixTrie Constructor

The RadixTrie class constructor takes two arguments, the first is a pointer to an OWEFArgs object, which represents the job options defined by the user, and the second is a pointer to a SequenceFile object which represents the input dataset. The constructor copies the OWEFArgs pointer to a private variable called list, which allows all RadixTrie object functions access to the input parameters without passing the pointer around. Similarly, the SequenceFile pointer is copied to a private variable for easy access. The RadixTrie class contains a pointer to a RadixTrieNode called root, which serves as the root node of the constructed trie. This pointer is initialized to NULL, and a variable used to track the size of the structure is set to zero (the rt_size variable). Once the base variables are set up, the countWords function is called to build the database. Following completion of building, the number of words contained and the size of the database are printed to standard out and, finally, the missSearch function is called, if applicable. The missSearch is used to search for nullomers, or words that are not contained in the database, and outputs any nullomers to a separate file.

5.2.2 The countWords Function

The main function used to build the trie is the countWords function, which takes the input dataset object and builds the radix trie structure with the data provided. For each individual sequence in the input, any filters required by the user are applied to potentially remove N's and ancestral repeats from the input. The resulting set of sequence fragments are then processed and all subwords up to the requested word length are added to the data structure using the incCount function. Listing 5.1 shows the pseudocode of processing the input and building the database. This code also contains the check for processing on a distributed system. If the framework is compiled for MPI distribution, the countWords function checks the first character of each processed word against its assigned prefix. If

they do not match, the word is not added to the data structure, if they do match, it is. This simple check allows the radix trie to process only a subset of the words based on a supplied prefix parameter.

```
for(i = all sequences)
   if(filters)
      perform filtering
   for(j = all resulting sequence fragments)
      for(k = all characters in fragment)
#ifdef KKURZ_MPI
         if(fragment[k] != list->pref)
            continue;
         else
#endif
            length = min( fragment length - k, maxlength );
            for(l = 1->length){
               char *t = fragment[k];
               incCount(k, l);
```

Listing 5.1: The countWords Function

5.2.3 The incCount Function

The incCount function is one of the most important, and simplest in the framework. It is used to check if a word already exists in the database before explicitly adding it to the structure. For each character in the input, the incCount function is called along with a length parameter to identify the length of the word being processed. First, the incCount function calls the trieGet function to see if the word already exists in the database, if it does, the trieGet function increments it as part of the search and the previous count value of the word is returned to the calling function. If the trieGet function returns (-1), the word

is not currently in the database and must be added. The trieAdd function is then called and the incCount function returns (1), indicating that the word has been added to the data structure.

5.2.4 The trieGet Function

The trieGet function is a recursive trie search function that has the side effect of incrementing the data for the search target when it is found. Starting at the root, the first character is mapped to an index in the branch pointer array and, if the pointer is not NULL, the function increments the the character pointer and decrements the length variable before calling itself recursively. This allows the function to search for arbitrary word lengths easily. When the length variable reaches zero, the node containing data for the searched word has been found and several other checks can occur. First, if the node's data variable is currently equal to zero, this word has not yet been found, so the num_words array of the OWEFArgs object must be updated to keep track of how many words are present in the database. Secondly, if the sequence currently being processed does not equal the last sequence stored on the node, this is the first time this word has been seen in this unique sequence, and the num_seq variable on the node needs to be incremented, and the last_seq variable must be updated to equal the current sequence. Finally, the count variable must be incremented and its previous value is returned to the calling function, indicating that the word was found and its value was incremented. If at any point the function reaches a NULL pointer in the trie traversal, the recursion ends and a value of (-1) is returned, indicating that the word does not exist in the database and must be added.

5.2.5 The trieAdd Function

The trieAdd function is very similar in logic to the trieGet function, with the exception of how it handles NULL pointers. The function is provided with a word that is

not contained in the trie, which it must add. trieAdd walks the trie until it reaches a NULL pointer, at which point it adds a new node and continues to travel the trie, stepping onto the new node. This behavior continues until the length variable reaches zero, indicating that the entire word has been added to the trie. Once the necessary nodes have been added, the node's status variables are initialized; the count and num_seq variables are set to one, and the last_seq variable is set to the current sequence being processed. The function then returns and processing the input may continue.

5.3 Searching the Structure

Once the database has been built, the real processing can begin. Subsequent stages in the framework's pipeline initialize an iterator for the data structure that allows them to systematically retrieve every word contained in the database. In addition, information about specific words may be requested using the supplied interfaces of the DataStructure parent class. This allows scoring stages to process each word, using frequencies of sub-words as statistics to create valuable scores for the words. The following subsections describe the radix trie specific implementations of the DataStructure class's standard interfaces and a description of the RtIterator class, which allows the retrieval of all words in the database. Table 5.3 provides an overview of the various functions used to search the radix trie.

Table 5.3: Functions for Searching the Radix Trie

Function name	**Parameters**	**Return values**	**Purpose**
getCount	string &motif	int	Return the number of occurrences of the motif in the database
getSeqs	string &motif	int	Return the number of unique sequences in which the motif occurs in the database
trieFind	RadixTrieNode * &node, char *s, int length	int	Return the number of occurrences of the motif in the database
trieFindS	RadixTrieNode * &node, char *s, int length	int	Return the number of unique sequences in which the motif occurs in the database

5.3.1 The RtIterator Class

The RtIterator contains only two functions and several state variables, but has one of the most important tasks in the framework. Objects of this class allow systematic access to all words in the database through the use of the hasNext and next functions (Table 5.4). The RtIterator class takes advantage of the num_words array in the OWEFArgs object that is populated by the RadixTrie class, and keeps track of how many words the iterator has returned internally. A comparison between the number of words that have been reported and the number of words present allows quick response from the hasNext function. Other stages must use the hasNext function prior to calling the next function, as the next function does not contain checks to keep it from traversing past the end of the trie. The hasNext function walks the trie via an in-order traversal and only returns when the word represented by the current location is of the length requested. If the iterator has previously returned a word, the next function uses the substring of the word, excluding the last character, as its starting position and continues its in-order traversal, returning the next word from that location.

Table 5.4: Functions for Searching the Radix Trie

Function name	**Parameters**	**Return values**	**Purpose**
hasNext	N/A	bool	True if there are more words, false otherwise
next	N/A	string	Return the next lexicographic word in the database

5.3.2 The getCount, getSeqs, trieFind and trieFindS Functions

These two functions work as one, as the getCount function is merely a wrapper around the trieFind function to provide the standard interface required by the DataStructure class. The getCount function takes the string representation of the word

being requested and passes a pointer to the first character of the word to the trieFind function, along with the length of the word in question. This allows the use of character math as an efficient way to walk the trie while still providing a simple string interface to the other stages of the pipeline. The trieFind function recursively travels the trie in the same way as the trieGet function, but eliminates the side effects introduced by that function, and simply returns the value of the count variable on the node without changing any of the information on the node.

5.4 Optimizations

During development, several optimizations were discovered to allow the RadixTrie class to be more efficient, both in processing time and space complexity. In Listing 5.1, the innermost for-loop runs from word length one up to the requested word length. In the worst case of an empty trie, the addition of the first word will add one node to the trie for each iteration of the loop, requiring both a lookup via the trieGet function and an addition via the trieAdd function. If the loop is inverted, however, and runs from the requested word length down to length 1, the shorter words are guaranteed to exist in the database, as the nodes for these sub-words have been added as part of adding the longest word. Then each shorter word only requires a call to trieGet to increment the count. This optimization does not save any space, but yields faster processing of the input data.

A second important optimization relative to access speed is in the translation from a character in the target word to a branch index for stepping through the trie. Originally, this required subtracting the character 'A' from the character in the word, then putting the result of the difference into a switch to determine which branch to follow (Listing 5.2).

```
int locateBranch( char x ){
    int branch_index = -1;
    x = toupper( x );
    char difference = x - 'A';
    switch(difference){
        case 'A':
            return 0;
            break;
        case 'C':
            return 1;
            break;
        case 'G':
            return 2;
            break;
        case 'T':
            return 3;
            break;
        default:
            return 4;
            break;
    }
}
```

Listing 5.2: Inefficient Branch Computation

This switch gets used hundreds of thousands of times during the processing of words, both in the addition of words to the data structure and during the retrieval of words from the database, and can be removed by pre-computing the branches for each character and placing them in an array (Listing 5.3 - locateBranch function). This is a constant lookup, helping improve the speed of the radix trie during building and lookups. A similar

mapping is required to translate the other direction; if branch x is followed, using that as an index into the reverse_branch array will provide the integer that must be added to 'A' to retrieve the character associated with that branch (Listing 5.3 - reverseBranch function).

```
static const int branch_array[ 26 ] =
   { 0, 4, 1, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4,
      4, 4, 4, 4 };
static const int reverse_branch[ 26 ] =
   { 0, 2, 6, 19, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
      13, 13, 13, 13, 13, 13, 13, 13, 13, 13 };

int locateBranch( char x ){
   x = toupper( x );
   return branch_array[x - 'A'];
}

char reverseBranch( int x){
   return 'A'+reverse_branch[x];
}
```

Listing 5.3: Optimized Branch Computation

One optimization that dramatically reduced the space requirements of the radix trie revolved around the array of RadixTrieNode pointers present on each node to point to the next node in the trie. For all leaf nodes, this array results in five NULL pointers per node, creating a large part of the space requirements that was unused and redundant. To solve this problem, the array of branch pointers was replaced with a pointer to an array of branch pointers. Each node is initialized as a leaf node, with its branches pointer set to NULL. When the node is extended, and becomes an internal node, the branches pointer is

modified to point to an array of five NULL-valued RadixTrieNode pointers, allowing those branch pointers to then point to the next nodes in the trie.

Other optimizations to the RadixTrie class are certainly possible, some requiring trade-offs that are too detrimental to include. One such idea involved storing additional statistics about the words, such as the scores computed by the word scoring classes, for all leaf nodes. In practice, however, these nodes proved to be too large to allow large datasets and long word lengths to be processed without thrashing the system. Currently, work is under way to remove all string variables from the framework, as the STL string libraries are notoriously inefficient, often making copies of entire strings just for small operations that can be done using character pointers. Other optimizations are being identified and assigned to team members as they become a priority.

6 Results

This chapter provides an overview of the possibilities of the OWEF framework as a flexible software system for bioinformatics. As an example of the quality and usability of the OWEF framework's design, two applications are described in section 6.1. Then, an in-depth look at the capabilities and limitations of the OWEF-based WordSeeker tool in its current form (Section 6.2.1) and a detailed look at the construction of the word landscape of the *Arabidopsis thaliana* genome, and its various segments, show the applicability of the framework to full-genome analysis (Section 6.2.2), while retaining validity with input of all sizes.

6.1 Applications of the OWEF Framework

An open-source project is worthless without a base of users who adopt its ideals and usage to drive its continued development. By default, the OWEF framework was adopted by the WordSeeker project, as it was built to aid the WordSeeker development process. It is important to note, however, that while WordSeeker is built on the OWEF framework, the framework is not WordSeeker. Rather, it provides a set of general interfaces that may be used by any tool that reads in a dataset, enumerates the words contained within and then performs some subsequent scoring of those words. In this section, the two projects currently utilizing the OWEF framework are discussed, and some additional possibilities are examined.

6.1.1 WordSeeker

In its early stages, the WordSeeker tool was proprietary software developed and used in-house at the bioinformatics lab at Ohio University. As research collaborations expanded, the decision was made to make the tool available to the public via the Ohio Supercomputer Center to garner increased usage. During this move, a redesign was

initialized, and the key components of the OWEF framework were identified. Once the basic framework was pieced together, the first child classes implemented were based on the existing algorithms and codebase of WordSeeker. The initial use of the framework as part of the WordSeeker project caused many unnecessary interfaces to be included in the parent classes for convenience, which were eventually edited out and narrowed down to the select few standard interfaces currently present in the OWEF framework. The following subsections discuss the various child classes implemented for the WordSeeker project under the OWEF framework.

6.1.1.1 DataStructure Class Children

This parent class has the most children implemented of any in the framework, as changes in the data structure used to store the database can drastically alter the framework's ability to process the input data up to the requested word length. Certain data structures are memory efficient, but computationally complex. For example, a simple data structure that would require almost no main memory would not build any explicit structure in memory, but would search through the input every time information was requested about a word. This would not be unfeasible for small input sets, but as the size of the input increases, the lookup time for each word would also increase dramatically. Some data structures, such as the radix trie, grow as the word length increases and remain more or less static in size for a specific word length, regardless of file size, while others, such as the suffix tree, become larger as file size increases, but have identical size on a given file, independent of the word length requested. This diversity in methodologies, makes the data structure a popular focus, and the RadixTrie, SuffixArray and SuffixTree classes are discussed below.

The RadixTrie Child Class

Originally implemented as a replacement algorithm for a previous version of WordSeeker, the RadixTrie class became an integral part of the early framework design and was a key contribution to this research. The radix trie is an extremely efficient way to store information about string input, as it allows linear lookup in word length for information about any word contained in the data structure [16]. The RadixTrie class is currently the default data structure used in the WordSeeker tool, as it has been the most tested and optimized of the DataStructure class children. The RadixTrie class is described in depth in chapter 5.

The SuffixArray Child Class

First introduced in [37], suffix arrays provide a significant improvement in space requirements over suffix trees, while still providing fast lookup of words. [37] describes an algorithm for constructing suffix arrays using space linear to the size of the input, roughly five times the length of the input, in bytes ($O(c * N)$). This data structure takes a significant amount of time to build, but can return all occurrences of a string presented to it in $O(L + \lceil log_2(N - 1) \rceil)$ time, where L is the word length and N is the length of the input. This performance is slower than the radix trie, but faster than the suffix tree for words with many occurrences, as the time to search is dependent on the word length and input length, rather than just word length (radix trie) or word length and number of occurrences (suffix tree). The SuffixArray class for the WordSeeker tool is currently under development by another member of the bioinformatics group at Ohio University, Lee Nau and results of testing this data structure will not be presented here.

The SuffixTree Child Class

Optimized versions of the suffix tree construction algorithm allow building the tree in $O(N)$ time, where N is the length of the input [38, 63, 67]. This data structure is also capable of returning all occurrences of a string in $O(N + k)$ time, where k is the number of occurrences of the word. Sagot presents in [51] a suffix tree for DNA alphabets that requires $O(n * N)$ space, where n is the average sequence length and N is the number of sequences in the input. It is easy to see that both the suffix array and the suffix tree grow as the size of the input file grows, a trait that becomes problematic as inputs move from small tests to full genomes. It should also be noted that, although the radix trie search time is capable of returning the number of occurrences of a word much faster than either the suffix array or suffix tree, the suffix-based approaches are at a disadvantage because they can also inherently provide location information as part of their results, which would have to be computed for the case of the radix tree. The implementation, testing and data collection for this structure were provided by Lev Neiman.

6.1.1.2 WordScoring Class Children

Currently, only one child class has been implemented for word scoring as part of the WordSeeker project, the MarkovModel class. This class provides statistical expected values for the number of times a word should be found in the input overall, and also how many unique sequences within the input should contain the word. Like the RadixTrie class, much of the code for this class was written prior to the first implementation of the framework, and was ported to work in the context of the OWEF interfaces.

The MarkovModel Child Class

The MarkovModel class computes expected values, and subsequent scores, for words based on the strategies outlined in [48] and [53]. Given a word length and Markov order,

the model computes the expected number of occurrences of a given word by gathering information about the occurrences of shorter subsequences of the target word. Additionally, since an input file may contain many unique sequences to be analyzed as a unit, the MarkovModel class computes a similar score for the expected number of unique sequences in which the target word should be found. Several additional scores are calculated to aid scientists in understanding the results; the $\frac{O}{E}$ score is the actual number of occurrences divided by the expected number of occurrences, and the $O * ln(\frac{O}{E})$ score is the number of occurrences multiplied with the natural logarithm of the $\frac{O}{E}$ score, both of which can provide insight into whether a word is over- or under-represented across the entire dataset based on total occurrences. The $S * ln(\frac{S}{E_s})$ score is the actual number of unique sequences multiplied with the natural logarithm of the number of unique sequences divided by the expected number of unique sequences, which ranks words by their coverage of the entire set of sequences, lowering the weight for words which occur multiple times within one sequence, but may miss many sequences.

In addition to these scores, a p-value calculation is provided to show that highly ranked words are statistically viable, and not simply occurring in high numbers by random chance. Also, the database may be checked for the reverse complement of the word (Figure 3.2) to identify interesting words on both strands of the DNA. Finally, the MarkovModel class provides access to a list of top ranked words for use by the ClusterMethod child classes and other post-processing stages.

6.1.1.3 ClusterMethod Class Children

Once scoring is complete, the next step in the framework's pipeline is word clustering, which groups together words of similar makeup. This is possible through the definition of word distances, based on metrics such as Hamming distance [24] or Levenshtein (edit) distance [30, 50], to build a cluster from words found in the input that

are within a user-specified distance. These two distances have been implemented as part of the WordSeeker toolkit, and are selected via a command line option at runtime.

The HammingCluster Child Class

For each seed word provided by the MarkovModel class, the HammingCluster class compares all words found in the database to determine if they are within the user-specified distance of each other. Hamming distance is described in [24] and defines the distance between two strings as the number of mutations required to transform one string into another (Figure 6.1). After creating a cluster, the HammingCluster class creates a motif out of the words contained in the cluster. Motifs have positions with variable characters, but usually also contain strong constrained subunits [14].

Seed word:	ACGTACGT
Comparison word:	CGTACGTA
Computing the Hamming Distance	
Mutation 1, C->A:	AGTACGTA
Mutation 2, G->C:	ACTACGTA
Mutation 3, T->G:	ACGACGTA
Mutation 4, A->T:	ACGTCGTA
Mutation 5, C->A:	ACGTAGTA
Mutation 6, G->C:	ACGTACTA
Mutation 7, T->G:	ACGTACGA
Mutation 8, A->T:	ACGTACGT
Total Hamming distance:	8

Figure 6.1: Computing the Hamming Distance Between Two Strings

The EditCluster Child Class

The logic of the EditCluster class is almost identical to that of the HammingCluster class, save the computation of the distance between any two words. Where Hamming

distance allows only mutations, and requires matching word lengths, edit distance allows any combination of insertions, deletions and mutations, and can compare words of different lengths. In its current form the EditCluster class always works with matching word lengths, but is capable of handling variable lengths [50]. Figure 6.2 shows the computation of the edit distance between two strings. The edit distance calculation is generally more flexible than that of Hamming distance, thus creating much larger clusters as output.

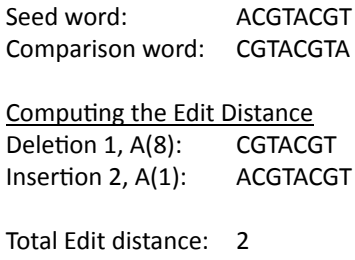

Figure 6.2: Computing the Edit Distance Between Two Strings

6.1.2 OpenMotif

The OpenMotif project began as a joint venture between the R'MES [27] group and the Ohio University bioinformatics lab to scale up the input size limits of the R'MES tool and improve the statistical models available in the bioinformatics tools from Ohio University. During preliminary discussions, many concepts were introduced, including modifying either R'MES or WordSeeker to incorporate the other tool's code. Eventually, as details were fleshed out, it became apparent that a new codebase, involving parts of both tools, was necessary and the OWEF framework provided an excellent basis for that collaboration. The key problem of the R'MES group revolved around the amount of time

needed to calculate statistics on a set of words, due to their inefficient data structure and word scoring methods. Since WordSeeker was already built on the OWEF framework, it was easy to provide OpenMotif with the existing RadixTrie class for highly optimized data storage and retrieval, eliminating one of the bottlenecks of the R'MES code. Next, the R'MES scoring model, also based on the principles of a Markov chain model, was brought in as a child of the WordScoring class. Additionally, R'MES contains a type of clustering which groups words into families based on ambiguous nucleotides placed at specific indexes within the word, which was also brought into the framework.

6.1.2.1 WordScoring Class Children

The OpenMotif project only contains one WordScoring child class, the RMESModel class, which is based on existing open-source code from the R'MES project. Development of this child class proceeded in two steps: first, the class was integrated into the framework with as few changes as possible made to the code, then the code was analyzed and optimized to be able to handle larger datasets in a reasonable timeframe.

The RMESModel Child Class

The RMESModel class, which is based off the code in the R'MES tool, is also based on a Markov model of the input sequence to provide expected occurrence levels for words in the dataset. This class also computes a variance measure to provide over- and under-representation information about the word in question. During testing, the R'MES tool was unable to process the *Escherichia coli* genome for word lengths greater than four, limiting its use as a research tool. While incorporating the tool into the OWEF framework as a child of the WordScoring class, two reasons for this were identified. First, the data structure being used was highly inefficient, leading to large main memory requirements, a problem easily solved by using the optimized RadixTrie class as the data structure for the OpenMotif project. The second problem involved how the R'MES code proceeded to

score words. Rather than requesting words iteratively from the data structure, the original codebase created every possible permutation for the requested word length, then requested the number of occurrences, creating a tremendous number of requests for words that did not occur in the input, especially for longer wordlengths. This was rectified by using a loop similar to the one described in Listing 3.1 as the basic loop logic for the RMESModel class, and overriding the computeScores function to call the appropriate scoring functions from the R'MES codebase.

6.1.2.2 WordClustering Class Children

The word clustering methods provided in the R'MES tool have a significantly different methodology of clustering when compared to the Hamming and Edit distance models. This provided a significant challenge when incorporating the WordFamily class into the framework.

The R'MES WordFamily Class

Where most clustering methods build clusters by comparing the distance between two words as a basis for inclusion in the cluster, the R'MES word family method takes a seed word and creates a new word containing ambiguous nucleotides in certain positions. The class then compares this new word against all words contained in the database and any words that match the seed based on regular expression matching are added to the family (Table 6.1).

Obviously, as the number of ambiguous nucleotides, n, rises, the number of possible matches increases exponentially in the order of 4^n. Typically, the number of N's in the word family seeds is limited to being less than three, creating clusters of up to 4^3, or 64 members. For large datasets, checking every word in the database against the seed word is highly inefficient, so a function was added to the DataStructure interfaces in the

Table 6.1: Pattern Matching in the Word Family Class

Seed Word	**Possible WordFamily Members**
ACGTNACTG	ACGTAACGT, ACGTCACGT, ACGTGACGT, ACGTTACGT
ACGNNACTG	ACGAAACGT, ACGACACGT, ACGAGACGT, ACGATACGT, ACGCAACGT, ACGCCACGT, ACGCGACGT, ACGCTACGT, ACGGAACGT, ACGGCACGT, ACGGGACGT, ACGGTACGT, ACGTAACGT, ACGTCACGT, ACGTGACGT, ACGTTACGT

OpenMotif project called getRegexMatches (Table 6.2) to allow the WordFamily class to poll the database for the entire family in one call. This is an optimization for the WordFamily class specifically, so it was not included as part of the framework's standard interfaces, but shows the extensibility of the framework design.

Due to the difference in how word families are created, the WordFamily class was originally built as a separate stage of the framework because of development time constraints. This class is currently being moved to be a child of the ClusterMethod class, as further examination has determined that the WordFamily class is capable of operating within the standard interfaces, and the addition of the getRegexMatches function does not alter the basic functionality of the ClusterMethod class in any way.

Table 6.2: The getRegexMatches function

Function name	**Parameters**	**Return type**	**Purpose**
getRegexMatches	vector<string>&matches, string regex	void	Populate the vector with all words in the database that match the provided regular expression

6.1.3 Further Extension Possibilities

Once the idea of extending the framework to incorporate new algorithms becomes more familiar, it is easy to see vast possibilities for creating a strong codebase that includes many different data structures, word scoring models and word clustering methods. This would require, and help bolster, widespread adoption within the bioinformatics community, but could also provide a tremendously powerful tool for bioinformatics research.

6.1.3.1 Extending the DataStructure Class

As the name implies, the Open Word Enumeration Framework was built specifically with general word enumeration pipelines in mind. In reality, however heuristic word discovery algorithms could also be built into the framework, as long as there is a well-defined algorithm for systematically returning each word in the set. This extension would allow a whole new class of tools to take advantage of the speedy development environment provided by the interfaces in the OWEF framework and open the door to new data structure selection criteria, based not only on space and time complexity, but also on quality of heuristics.

6.1.3.2 Extending the WordScoring Class

The Markov model is the most common statistical model for computing the expected number of occurrences of a word in a biological dataset, but it is far from the only model for this task. [48] alone discusses normal and compound Poisson and Gaussian approximations for the distribution of a word in the input dataset, an extension of the expected count of the word. Ideally, every known model would eventually be implemented as a child of the WordScoring class, allowing the user to work with unlimited options for computing statistics about the words found in their dataset. As new

methods are developed or adapted, these could quickly be added to the existing models and allow easy comparison of each model's quality against others. When used in conjunction with a single enumerative data structure, the rest of the experimental setup can be fixed, allowing true analysis of one model against another.

6.1.3.3 Extending the ClusterMethod Class

Similar to the idea of creating a superset of all word scoring models into the framework, one could also aspire to create child classes for many different word clustering methods. As many clustering methods differ only by the distance metric used for inclusion or exclusion of a word from a cluster, it may be possible to lump many of these methods into a single class and select the metric dynamically. Methods like the R'MES word family clusters will require unique classes, but can still fall under the general parent ClusterMethod class.

6.1.3.4 Extending the Shell Classes

Currently, the shell classes contain little code, making them hard to extend. As developers begin to implement sequence clustering algorithms, for example, the standard interfaces needed by these algorithms should emerge and development of a parent class can begin, and multiple methods can be present in the codebase, allowing informed decisions to be made about which technique to use. These new developments will help the OWEF framework remain relevant and useful.

6.2 WordSeeker - A Use Case

In this section, the WordSeeker tool is presented as an example use case of the OWEF framework. First, the current physical limitations of the tool are examined, with respect to both input file size and word length of the desired output. The various levels of parallelism that have been implemented are tested for performance and speedup. Once the

limits have been identified, a full genomic study of the *Arabidopsis thaliana* genome is presented as a biological application. *Arabidopsis*' use as a model plant genome has fostered many projects, including databases of information [15, 41, 61], hundreds of journal articles and several case studies of the genome [2, 36, 39].

6.2.1 Pushing the Limits of the WordSeeker Tool

A key concept of the framework is the ability to handle large inputs and process long word lengths. To this end, the WordSeeker tool was given inputs of various lengths and parameters to test its processing limitations. Due to the comparatively long run time of the scoring stages (Figure 6.3) and their negligible contribution to memory complexity, analysis was done with only the enumeration of words for two of the three implemented data structures. The motivation for this decision was to provide a rough guide to the word lengths and input sizes that the tool can handle (Section 6.2.1.1). In addition to this study, data was collected to show the speedup of the WordSeeker tool using both the shared and distributed memory approaches (Section 6.2.1.2).

These four sequences vary in length from 3.8 MB for the core promoter sequences up to 116 MB for the full *Arabidopsis* genome, and provide a good range of both file size and complexity, as the three smallest files contain short sets of sequences for each gene in the genome, or the genes themselves in the case of the coding sequences, while the full genome-wide file contains each of the five chromosomes of *Arabidopsis thaliana* as a sequence. It is obvious from Figure 6.3 that the full genome analysis of *A. thaliana* has enumeration times that far outweigh the scoring and clustering times. Upon further inspection, it can be seen that the enumeration times are proportional to the enumeration times of the smaller datasets, and the dramatic reduction in scoring time can be attributed to the fact that the genome-wide input contains only five unique sequences, while the smaller files each contain more than 27000 sequences, one for each gene in the

Arabidopsis thaliana genome. The computation of an expected number of sequences for each word in the genome-wide input is far less computationally intense than the same computation on an input containing thousands of sequences.

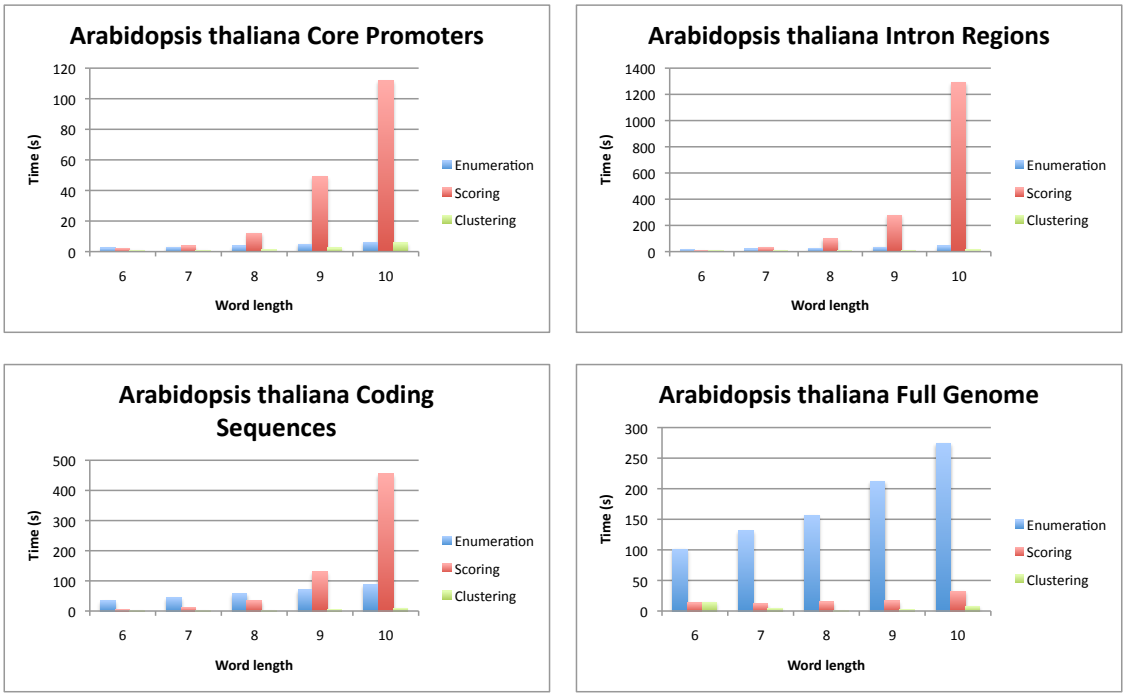

Figure 6.3: Execution Time Breakdown by Stage

6.2.1.1 Upper Boundaries

Previously, it was discussed that the suffix tree and suffix array space complexities are directly related to input file size, as they store indexes into the original file and must store each occurrence of a word individually. Additionally, the space requirements of the radix trie were discussed, both in theory and in practical application. In this section, a view of the input file size and word length restrictions of the radix trie and suffix tree

implementations is presented (Table 6.3), followed by a discussion of how this affects run time decisions on the selection of a data structure.

Table 6.3: Maximum Word Length and File Size for Radix Trie

Input File	**Size** (MB)	**Maximum Word Length**	
		Radix Trie	**Suffix Tree**
A. thaliana Core Promoters	3.8	100	100
A. thaliana Intron Regions	25	65	10234
A. thaliana Full Genome	116	20	N/A
H. sapiens Full Genome	3095	15	N/A

Both data structures are capable of processing every word in the core promoters dataset, as each sequence included is 100 characters in length, making length 100 the maximum. The radix trie is able to process the intron regions up to length 65, well past the normal length of regulatory elements, and the suffix tree is able to process words of up to the length of the longest individual sequence, which is more than 10000 characters long. The full genome of *Arabidopsis thaliana* is much longer than the intron regions, and proves to be too large for the suffix tree to process without thrashing the system, while the radix trie is capable of processing words of up to length twenty, which is again long enough to find regulatory elements in the dataset. The human genome is included in this test to provide the possibility of comparison to other tools, as this input is the ”holy grail” of bioinformatics today, and no tools have yet been able to process long word lengths in main memory [25]. Interestingly, the distributed radix trie implementation is capable of processing word lengths past the normal lengths of eukaryotic regulatory motifs, which could make it useful in the search for statistically interesting motifs in the *Homo sapien* genome. As expected from the literature [16, 25] and smaller tests, the suffix tree was unable to process the human genome file.

6.2.1.2 Speedup of the WordSeeker Tool

Figures 6.4 and 6.5 show the speedup and efficiency of the shared memory with various numbers of processor cores and for the 8 core shared memory versus the distributed system for both the radix trie and suffix tree. To compute the speedup for N processors, the formula $S_N = \frac{T_s}{T_p}$ is used, where T_s is the time for the serial algorithm and T_p is the time for the parallel algorithm. The efficiency, E_N, is the speedup, S_N, divided by the number of processors used, N. This is shown by the formula $E_N = \frac{S_N}{N}$.

The performance of the radix trie in the speedup measurements indicate two important pieces of information about the implementation. First, given enough words in the dataset, the parallelization of the scoring routines can have a considerable impact on the amount of time spent scoring the information in the trie. The input in this case, the *Arabidopsis thaliana* core promoter sequences, is a small dataset with short sequences, causing a drop in speedup performance for both short words, as there are few possible, and long words, as there are only a few of the possible words occurring in this dataset (Figure 6.4(a)). From length ten to fifty, the tool performs well, but remains shy of linear speedup.

This less than ideal performance can be attributed to the second piece of information that can be gleaned from these plots. Assuming enough words occur in the dataset to overcome the cost of setting up the parallel environment, as the number of cores increases, the use of inefficient locking schemes lowers the performance, causing the efficiency to drop off significantly for 8 cores (Figure 6.4(b)). In its current implementation, the MarkovModel scoring stage of the WordSeeker tool locks large sections of code to perform sequential updates to critical variables and avoid deadlocks. Additional development is needed to reduce the size of sequentialized blocks of code and increase performance for the shared memory version of the OWEF framework.

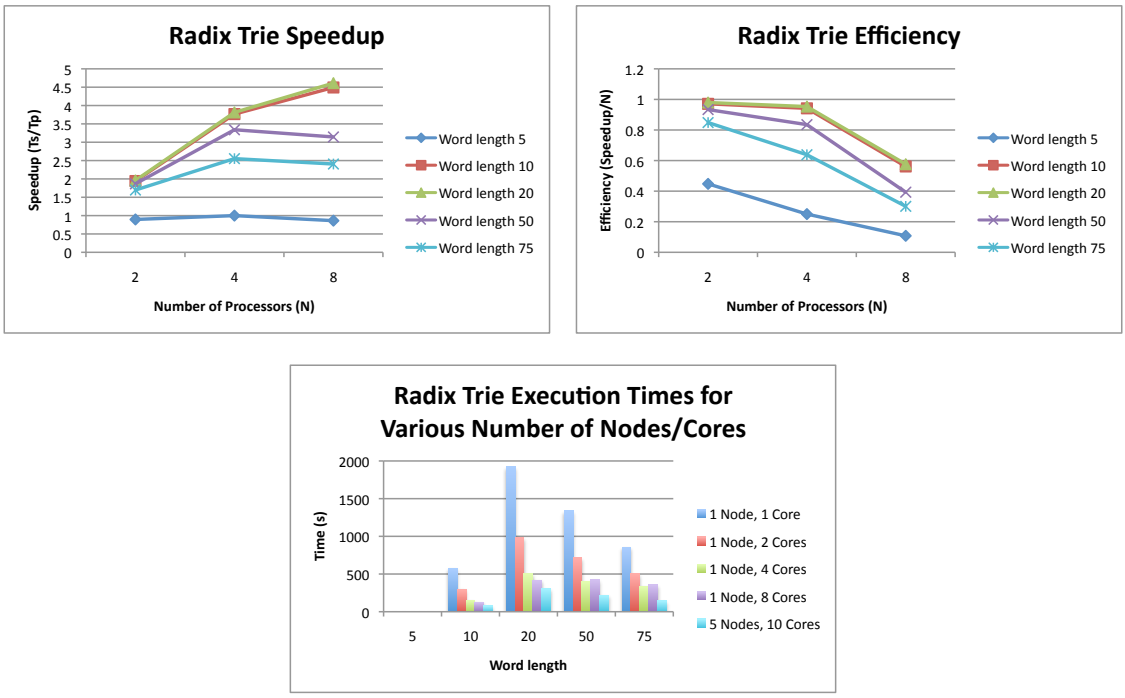

Figure 6.4: Performance Analysis of the Radix Trie

The plot of the overall execution times for the various word lengths (Figure 6.4(c)) brings another valuable piece of information about the performance of the WordSeeker tool: using the most powerful computing configuration available will always yield the fastest runtimes. This may seem obvious, but distributed systems often run into network communication bottlenecks, causing them to perform poorly on certain inputs. Originally the distributed version of the framework was designed to alleviate main memory bottlenecks, but for the WordSeeker tool using the RadixTrie class as a data structure, the benefit is two-fold as execution times improve as well.

The suffix tree performed similarly to the radix trie during speedup and efficiency testing, as the difference between Figures 6.4 and 6.5 can be attributed, at least partially, to the additional time cost for building the entire suffix tree on the input. Interestingly, the

Figure 6.5: Performance Analysis of the Suffix Trie

overall runtimes of the suffix tree are significantly faster than those of the radix trie, with length 20 only running for about half as long. This is most likely due to recent optimizations in the SuffixTree class to eliminate string variable and instead replace them with pointers to Sequence objects. This has shown strong improvement in runtimes, but has not been fully ported to the rest of the framework. Also, if all words occur only a few times, the penalty usually incurred by the suffix tree when counting occurrences will be minimized, again cutting into the radix trie's theoretical advantage.

Figure 6.4 was created using the table listed in Tables 6.4 and 6.6. Figure 6.5 was created using the information in Tables 6.5 and 6.7. Shared memory tests were performed on a 64-bit Linux machine with 4 Dual-Core 2.6 GHz AMD Opteron processors and 32 GB of main memory. Distributed memory tests were performed on a 64-bit Linux

machine with 4 Quad-Core 2.5 GHz AMD Opteron processors and 24 GB of main memory.

Table 6.4: Speedup and Efficiency for Various Cores Using Radix Trie

Word Length	2 Cores		4 Cores		8 Cores	
	Speedup	Efficiency	Speedup	Efficiency	Speedup	Efficiency
5	0.8952	0.4476	1	0.25	0.8624	0.1078
10	1.9422	0.9711	3.7666	0.9416	4.4920	0.5615
20	1.9593	0.9796	3.8127	0.9531	4.6110	0.5763
50	1.8669	0.9334	3.3402	0.8350	3.1436	0.3929
75	1.6971	0.8485	2.5527	0.6381	2.4057	0.3007

Table 6.5: Speedup and Efficiency for Various Cores Using Suffix Tree

Word Length	2 Cores		4 Cores		8 Cores	
	Speedup	Efficiency	Speedup	Efficiency	Speedup	Efficiency
5	0.9902	0.4951	1.0048	0.2512	1.0052	0.1256
10	1.7123	0.8561	2.6753	0.6688	3.1779	0.3972
20	1.8068	0.9034	3.0401	0.7600	3.4534	0.4316
50	1.5294	0.7647	2.1245	0.5311	2.2975	0.2871
75	1.2691	0.6345	1.4742	0.3685	1.4955	0.1869

Table 6.6: Total Runtimes for Radix Trie

Word Length	1 Core	2 Cores	4 Cores	8 Cores	Distributed
5	1.99	2.22	1.99	2.31	0.64
10	575.95	296.53	152.90	128.21	78.55
20	1922.06	980.99	504.11	416.84	301.58
50	1337.74	716.55	400.49	425.54	216.03
75	856.31	504.56	335.45	355.94	143.91

Table 6.7: Total Runtimes for Suffix Tree

Word Length	1 Core	2 Cores	4 Cores	8 Cores	Distributed
5	34.92	35.27	34.75	34.74	14.72
10	272.81	159.32	101.97	85.84	93.33
20	827.63	458.05	272.23	239.65	321.71
50	633.71	414.35	298.28	275.82	242.54
75	480.70	378.76	326.06	321.42	173.27

6.2.2 Word Landscape Analysis of *Arabidopsis thaliana*

A model organism used for the study of plant biology, the *Arabidopsis thaliana* plant provides a relatively simple biological system through which researchers can learn about the intricacies of stress response [70], binding site regulation [40] and many other important functions. *Arabidopsis* is a flowering plant that often shows changes in its genome in a highly visible way, easing the study of these modifications. A study performed by Lichtenberg et al. in 2009 [36] specifically studied all words of length 8 in the non-coding regions of the genome, for this study, that concept was extended to analyze all available segments of *Arabidopsis* for words of length one through twenty. The lower boundary was chosen because it provides a basic level of knowledge about the makeup of the *Arabidopsis thaliana* genome, showing what percentage of the genome is encompassed by each different nucleotide. The upper limit was chosen for two reasons, primarily, the decision was easy because length twenty is the longest length the current version of the tool can process for the full genome (Table 6.3). Secondly, as shown in Figure 6.6, length twenty and below analysis will identify almost 95% of the binding sites currently known for *Arabidopsis*.

Using the WordSeeker tool built on the OWEF framework, the various segments and full genome of the *Arabidopsis thaliana* genome were analyzed from word length one to twenty, providing full statistical analysis of *A. thaliana*. Tables of the top twenty-five

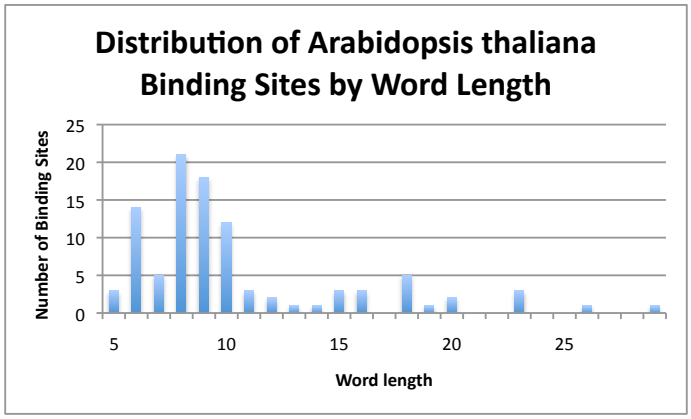

Figure 6.6: A Histogram of *Arabidopsis thaliana* Binding Sites by Word Length

scored words sorted the $S * ln(\frac{S}{E_s})$ score are shown in appendix A, and the full output files for each segment and word length are available (http://bio-s1.cs.ohiou.edu/~kkurz/ThesisDatasets). Words of length one through twenty were chosen because this set of lengths easily encompasses the typical length of regulatory elements, which are often between length six and eight [62], and several longer motifs have been reported for *Arabidopsis* in the Arabidopsis Gene Regulatory Information Server, or AGRIS database[15, 41]. The following sections give a brief overview of each segment's analysis, providing times and data structure size measurements along with a short description of the results. All testing in this section was performed using the RadixTrie class as the data structure, the MarkovModel class as the word scoring model and the HammingCluster class as the clustering method. For simplicity, all tests except for the *Arabidopsis thaliana* full genome, lengths 18, 19 and 20 were performed on a shared memory architecture, as described above. Lengths 18, 19 and 20 for the full genome were unable to finish on the shared memory system, so those runs were performed on the distributed system. The top five words sorted by $S * ln(\frac{S}{E_s})$ are shown for word lengths 6, 8 and 10 below, with the exception of the full genome analysis, which is sorted by $O * ln(\frac{O}{E})$,

since the full genome input file contains only five unique sequences, one for each chromosome. All AGRIS lookups were performed using [71]. A similar analysis of *Arabidopsis* was performed in [36], excluding the coding regions, with a previous version of the WordSeeker tool, prior to full integration with the OWEF framework.

6.2.2.1 The *Arabidopsis thaliana* 5' Untranslated Regions

The 5' UTR regions contain the genomic data immediately downstream of the core promoter sequences and immediately upstream of the coding sequences, and have been shown to contain interesting protein binding sites in [33, 54]. Table 6.8 shows the statistics about the tasks used to analyze this dataset, and Tables 6.9, 6.10 and 6.11 show the top scored words for this dataset.

Table 6.8: Size and Timing Information, 5' UTR

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	2.299695	2.917490	1.121598	5.217185
8	6	0.002958	3.590521	19.253780	1.416932	22.844301
10	8	0.035570	6.154461	157.014079	5.855332	163.16854

As Table 6.8 shows, the 5' UTR regions of the *Arabidopsis* genome are relatively simple for the WordSeeker tool to process, with even length 10 running in under three minutes, and the database remaining under 35 MB out of the available 24 GB.

Table 6.9: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, 5' UTR

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTCTTT	3608	2979.93	4622	3623.72	1.27549	1124.66	690.041
ACCCTA	1294	763.683	1381	824.08	1.67581	713.004	682.379
TTCTCC	2910	2302.65	3391	2696.69	1.25747	776.874	681.214
CAAAAC	2316	1869.71	2643	2138.81	1.23573	559.426	495.753
TAGGGT	1155	777.041	1268	839.075	1.51119	523.552	457.798

Table 6.10: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, 5' UTR

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTCTTCTC	877	620.404	999	675.022	1.47995	391.618	303.561
CTTTCTCT	1173	1021.67	1314	1135.35	1.15735	192.023	162.024
AACAAAAA	1059	929.997	1142	1028.48	1.11038	119.57	137.563
TTTCTTCA	617	497.591	637	537.934	1.18416	107.675	132.711
GAGAAGAG	320	212.597	365	226.456	1.61179	174.232	130.856

Table 6.11: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, 5' UTR

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTCTTCTTT	192	108.694	192	116.959	1.64159	95.1682	109.24
CTCTCTCTTT	532	444.414	552	486.735	1.13409	69.4568	95.6995
AAAGAAGAAA	156	89.0452	157	95.7178	1.64024	77.6902	87.4711
CTTTCTCTCT	455	388.406	471	424.137	1.11049	49.362	72.0018
ACTCTCTCTT	115	66.3207	116	71.2056	1.62908	56.6101	63.2995

The top five words of the 5' UTR for lengths 6, 8 and 10, sorted by $S * ln(\frac{S}{E_s})$ are shown in Tables 6.9, 6.10 and 6.11. The AGRIS [15, 41] database was queried with each of these words and, while none of these words matches a known binding site, several words, CTCTTCTC and AAAGAAGAAA in particular, have repetitive structure, making them intriguing for possible additional examination.

6.2.2.2 The *Arabidopsis thaliana* 3' Untranslated Regions

A companion to the 5' UTR, the 3' Untranslated Regions are regions immediately following gene coding sequences that are not translated into protein during protein synthesis. Although not adjacent to typical promoter regions, the 3' UTR has important regulatory characteristics in eukaryotic cells [68], and is useful in the study of the full word landscape of *Arabidopsis*.

Table 6.12: Size and Timing Information, 3' UTR

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	4.092285	3.004922	1.404120	7.097207
8	6	0.002961	6.286397	21.548922	1.499570	27.835319
10	8	0.038819	10.780911	201.141277	6.734395	211.922188

The timing and database size information for the 3' UTR is similar to that of the 5' UTR, although the 3' UTR for any given gene tends to be slightly longer than the 5' UTR, leading to the somewhat longer execution time and larger database size, as more of the possible words at the selected length are likely to occur (Table 6.12).

Table 6.13: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, 3' UTR

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTG	4755	4160.29	5682	4989.47	1.1388	738.514	635.328
GTTTTG	5147	4589.77	6377	5602.8	1.13818	825.38	589.765
ATTTTG	5320	4781.61	6432	5883.9	1.09315	572.875	567.617
ATTTTA	4395	3863.93	5257	4578.6	1.14817	726.344	565.998
TAAAAT	3757	3238.25	4337	3742.48	1.15886	639.419	558.247

Table 6.14: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, 3' UTR

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTGTT	2292	2092.08	2520	2334.61	1.07941	192.56	209.182
TTTTTTGG	1010	832.111	1060	885.402	1.1972	190.782	195.677
TTTTTCTT	2192	2005.96	2428	2231.13	1.08824	205.311	194.416
ATTTTGTA	739	589.001	759	621.081	1.22206	152.21	167.657
TAATTTTT	794	647.449	817	684.196	1.1941	144.932	162.011

Table 6.15: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, 3' UTR

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTCTTTTTT	270	207.593	273	217.821	1.25332	61.6425	70.9676
GATTTTTTTT	232	186.826	234	195.88	1.19461	41.6091	50.2416
GTTTTTTTTT	744	695.852	777	743.532	1.04501	34.2107	49.7765
TTTTTTTCTT	666	618.872	689	659.377	1.04493	30.2783	48.8788
AGTTTTTTTT	339	295.55	346	311.126	1.11209	36.7595	46.4981

Tables 6.13, 6.14 and 6.15 show the top five words for lengths 6, 8 and 10, and each word was submitted to AGRIS as before. Again, none of the words presented here match known binding sites. It is interesting to note that most of the words in these tables contain strings of T's, helpful for the post-transcriptional polyadenylation of mRNA.

6.2.2.3 The *Arabidopsis thaliana* Core Promoters

These short sequences represent the one hundred base pairs immediately upstream of the transcription start site (TSS), where the RNA polymerase II-binding sites are typically found. The limited length of the core promoters does not fully manifest itself in the 6, 8 and 10 word length output, but comparing Table A.43 to Table A.64 shows exactly how the average sequence length can affect the size of the database. Where each core promoter sequence is one hundred nucleotides, the proximal promoters are each nine hundred base pairs, creating a much larger input file and allowing more of the possible words to occur in the input for each word length. For length 20, the database for the core promoters requires only around 1.25 GB, while the storage of the same word length for the proximal promoters requires more than 10 GB!

Table 6.16: Size and Timing Information, Core Promoters

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	2.408993	2.151728	0.807213	4.560721
8	6	0.002962	3.704660	12.161700	1.098893	15.86636
10	8	0.036175	6.160703	111.959462	5.726779	118.120165

Word lengths less than 10 are too short to see large differences in database size due to file size difference, as shown in Table 6.16. The bulk of processing time is encompassed by the scoring stage, which directly queries the database for information, yielding similar runtimes for the core promoters when compared to the 5' and 3' UTR segments.

Table 6.17: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, Core Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATAAA	5284	4351.25	5675	4987.01	1.13796	733.401	1026.26
TAAAAT	3421	2726.79	3822	3023.08	1.26427	896.25	775.914
CAAAAC	2740	2089.77	2984	2287.99	1.3042	792.52	742.282
AATATT	2664	2057.81	2869	2251.6	1.2742	695.217	687.81
TAAATA	3783	3185.49	4195	3564.24	1.17697	683.544	650.336

Table 6.18: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, Core Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATAAATA	1373	1085.42	1387	1190.81	1.16476	211.533	322.7
CTATAAAT	726	480.418	730	521.15	1.40075	246.015	299.76
CTATATAA	640	414.007	642	448.558	1.43125	230.189	278.774
ATATAAAC	562	351.539	562	380.439	1.47724	219.28	263.68
TAAAAAAT	477	298.661	484	322.899	1.49892	195.896	223.335

Table 6.19: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, Core Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTATATATAC	122	67.7584	122	74.5516	1.63645	60.0886	71.7449
TAAAAAAAAT	96	50.3785	97	55.4117	1.75053	54.3122	61.8993
CTATAAATAC	181	128.934	181	142.019	1.27448	43.8994	61.3949
TATAAATAAG	99	54.9764	99	60.474	1.63707	48.7977	58.2334
AAAAAAAAAG	514	461.754	528	511.735	1.03178	16.521	55.0959

As before, all words in Tables 6.17, 6.18 and 6.19 were submitted to AGRIS for comparison against known binding sites. And again, none of the highest ranked words matched the binding sites stored in the AGRIS system. Interestingly, the TATA-box (TATAAA) is seen in all three tables, but was not reported as a known binding site by AGRIS, despite its known importance to RNA-polymerase II-transcribed genes [58]. From these tables, the core promoters seem to have a much higher A/T content than G/C content, which is confirmed by Table A.44.

6.2.2.4 The *Arabidopsis thaliana* Proximal Promoters

Together, the core and proximal promoters cover the region one thousand base pair upstream of a gene, with the core promoters being the closest one hundred nucleotides and the proximal promoters being the additional nine hundred. This large region commonly contains additional regulatory elements to augment the main transcription factor binding sites that control base level gene expression.

Table 6.20: Size and Timing Information, Proximal Promoters

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	21.011225	6.202330	2.462631	27.213555
8	6	0.002971	34.704883	17.074537	1.756133	51.77942
10	8	0.046703	57.245912	217.210981	8.470263	274.456893

The proximal promoters form a considerably larger input file than any of the previously discussed genomic segments, leading to a slightly longer runtime and larger database (Table 6.20) for word lengths where all the smaller segments do not contain all possible words, usually this begins around length 8 for small inputs from the *Arabidopsis* genome.

Table 6.21: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, Proximal Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAC	13854	12339.6	20348	16536.2	1.23051	4220.79	1603.79
GTTTTG	13059	11641.8	18878	15280.8	1.23541	3990.81	1500.18
TGAAAT	13301	12352.3	18824	16559.7	1.13674	2412.53	984.241
TAGAAT	8488	7592.08	10602	8952.1	1.1843	1793.38	946.822
GATAAG	5565	4702.47	6423	5191.78	1.23715	1366.87	937.192

Table 6.22: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, Proximal Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAAAAAAT	4282	3436.7	4873	3702.85	1.31601	1338.15	941.645
ATTTTTTA	3911	3167.35	4419	3393.88	1.30205	1166.35	824.818
TTATATAA	3107	2521.71	3405	2667.13	1.27665	831.645	648.491
AATATATT	3652	3127.75	4112	3348.74	1.22793	844.301	565.921
GAAAAAAG	2076	1660.65	2192	1726.94	1.2693	522.712	463.436

Table 6.23: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, Proximal Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTA	1036	690.564	1065	706.547	1.50733	437.012	420.216
TAAAAAAAAT	1078	748.72	1117	766.887	1.45654	420.062	392.929
AATATATATT	739	445.706	756	453.938	1.66543	385.621	373.666
TTATATATAA	541	320.103	558	325.254	1.71558	301.182	283.904
CATATATATG	313	150.473	316	152.415	2.07328	230.406	229.247

The proximal promoters contain one known binding site retrieved from AGRIS, the word GATAAG (Table 6.21) is known as the I box promoter motif and is reported in [20]. The I box is a promoter for a subunit of an important factor in photosynthesis, ribulose 1,5-bisphosphate carboxylase/oxygenase, which aids in CO_2 fixation. No other known sites were identified in the top words of these segments (Tables 6.21, 6.22 and 6.23).

6.2.2.5 The *Arabidopsis thaliana* Distal Promoters

Made up of the two thousand nucleotides upstream of the proximal promoters, the distal promoters are meant to contain any regulatory elements for a gene not already found in the proximal and core promoters. If two genes are located close to one another, the distal promoter for the downstream gene may contain part of the upstream gene's 3' UTR and coding sequences.

Table 6.24: Size and Timing Information, Distal Promoters

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	46.499415	7.982420	3.748664	54.481835
8	6	0.002971	74.714318	23.299167	1.878490	98.013485
10	8	0.047409	128.083779	224.766201	8.761543	352.84998

The database for the distal promoters is almost exactly the same size as the database for the proximal promoters, despite the input file being more than twice as large. This is attributed to the radix trie's dependence on the number of words found, rather than the length of the input. For length 10, almost all possible words were found in the proximal promoters, and those few missing words were found in the distal promoters, accounting for the small increase in size.

Table 6.25: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, Distal Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAATTG	13193	11983	18664	15842.1	1.17813	3059.51	1269.17
CATATC	12451	11267.2	17274	14588	1.18413	2919.41	1243.89
CAATTC	13120	11934.7	18603	15755.8	1.18071	3090.25	1242.25
TAGAAT	14113	12946	20711	17626.2	1.17501	3340.25	1218.07
GATAAG	11231	10112.8	15126	12679.5	1.19295	2668.67	1177.84

Table 6.26: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, Distal Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTA	5809	4887.98	7224	5407.46	1.33593	2092.29	1002.81
TAAAAAAT	5897	5014.08	7353	5562.19	1.32196	2052.34	956.455
GAAAAAAG	3583	2836.49	3925	3006.2	1.30563	1046.75	837.104
TTATATAA	4822	4128.99	5704	4494.25	1.26918	1359.66	748.158
CTTTTTTC	3551	2888.7	3907	3064.76	1.27481	948.619	733.008

Table 6.27: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, Distal Promoters

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTA	1520	1037.56	1606	1062.67	1.51129	663.223	580.393
AATATATATT	1166	724.697	1229	737.855	1.66564	627.047	554.526
TAAAAAAAAT	1475	1039.15	1555	1064.33	1.46102	589.551	516.624
TTATATATAA	944	569.023	974	577.663	1.6861	508.837	477.859
AAAAAAAATC	1922	1645.05	2029	1704.61	1.1903	353.47	299.059

The distal promoters also contain the same I box binding site as the proximal promoters (Table 6.25), with no other known binding sites being identified by AGRIS, although the TATA-box and some poly-A repeats can be seen again in Tables 6.26 and 6.27, indicating that some short genes may be fully contained in these long upstream regions, as well as the 3' UTR of other genes.

6.2.2.6 The *Arabidopsis thaliana* Coding Sequences

These sequences were not included in [36], but were included in this study from a desire to analyze the entire *Arabidopsis* genome by segment. The coding sequences consist of all identified genes in the genome on both the forward and reverse strands. There are approximately twenty-seven thousand gene coding sequences in the *Arabidopsis thaliana* genome.

Table 6.28: Size and Timing Information, Coding Sequences

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	28.603787	4.760546	2.872546	33.364333
8	6	0.002971	47.330480	34.618181	2.214564	81.948661
10	8	0.047402	78.750148	454.900436	9.520215	533.650584

The now familiar content in Table 6.28 shows the runtimes and database sizes by word length for the coding sequences input file. Word length 10 has an inordinately long runtime, possibly due to another process being run concurrently on the system when this task was running, as the two shorter word lengths have runtimes close to what would be expected based on the previous segment analyses.

Table 6.29: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, Coding Sequences

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATGGCG	7238	6248.58	8680	7645.37	1.13533	1101.68	1063.92
GAGTAT	7629	6646.43	9966	8247.77	1.20833	1885.94	1051.86
GGGTTT	9519	8562.14	13735	11410.8	1.20368	2546.23	1008.44
TAATAA	5043	4135.41	6423	4710.75	1.36348	1991.38	1000.6
TAGTAG	4030	3178.31	5042	3511.2	1.43598	1824.43	956.789

Table 6.30: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, Coding Sequences

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ACTTCTTG	1317	1008.36	1379	1044.45	1.32032	383.184	351.682
TAAGATTG	956	684.288	983	702.056	1.40017	330.874	319.667
AAAGAAGG	2440	2148.91	2672	2303.39	1.16003	396.646	309.969
TGAGATTG	2052	1765.99	2232	1871.05	1.19291	393.719	308.015
CTCTTCTC	2352	2069.68	2598	2213.11	1.17391	416.572	300.752

Table 6.31: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, Coding Sequences

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TGAAGAAGAT	869	694.208	900	713.64	1.26114	208.814	195.153
TTGATGATGT	311	197.685	315	200.301	1.57263	142.616	140.92
CTTCTTCCTC	710	588.539	739	603.145	1.22524	150.12	133.211
GATGAAGAAG	1236	1114.97	1315	1160.49	1.13314	164.37	127.378
GGAGAAGAAG	1126	1007.19	1193	1044.97	1.14166	158.049	125.554

All words in Tables 6.29, 6.30 and 6.31 were again submitted to AGRIS for binding site identification, but none of the words matched known binding sites. This was expected for these sequences, as most regulatory element studies focus on non-coding sequences. Notably, several of the words in these tables include sub-words representing either the START-codon (ATG) or one of the STOP-codons (TAA, TAG, TGA) used to identify the beginning or end of RNA polymerase II-transcribed genes.

6.2.2.7 The *Arabidopsis thaliana* Intron Segments

These segments represent the sections of genes that are often removed during protein translation, allowing multiple proteins to be encoded by a single gene. Several studies have identified regulatory elements in both *Arabidopsis* [29] and other genomes [8, 47].

Table 6.32: Size and Timing Information, Intron Segments

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	16.838000	8.426837	6.760218	25.264837
8	6	0.002969	26.261253	98.301754	6.008992	124.563007
10	8	0.043770	44.300981	1288.922963	12.824636	1333.223944

Table 6.32 contains the timing information for the analysis of the intronic segments of the *Arabidopsis thaliana* genome. Since each gene may have multiple introns, the number of sequences in the introns is much greater than the number of sequences in any of the other segment inputs, causing computation of the E_s score to rise significantly, leading to much longer runtimes than the previous segments.

Table 6.33: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6, Intron Segments

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTAAGT	13240	9743.29	13661	11010.7	1.24071	2946.42	4060.23
TTGCAG	15359	12102.5	15921	13977.5	1.13904	2072.74	3659.93
TTTCAG	15849	12627.6	17220	14654.8	1.17504	2777.63	3601.24
GTTTTG	23453	20458.4	29423	25531.7	1.15241	4173.87	3203.8
TGTTTT	38710	36070.6	53508	52307.9	1.02294	1213.79	2733.67

Table 6.34: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8, Intron Segments

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTGTT	9926	9245.61	10963	10545.2	1.03962	426.002	704.837
TTTTTCTT	9086	8422.61	9952	9532.87	1.04397	428.215	688.857
CTTTTTTC	2742	2154.45	2799	2298.27	1.21787	551.703	661.242
GTTTTTGA	2643	2073.32	2711	2210	1.2267	553.926	641.623
TTTTGCAG	3479	2936.05	3497	3155.58	1.10819	359.256	590.319

Table 6.35: Top 5 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10, Intron Segments

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTA	608	406.863	614	432.262	1.42043	215.491	244.233
GATTTTTTTT	1203	990.289	1222	1058.21	1.15478	175.857	234.076
TTTTTGTGTG	604	436.816	607	464.224	1.30756	162.774	195.733
GTTTTTTTTT	2930	2742.44	3048	2981.18	1.02241	67.5632	193.831
TTTTTTTTGT	2614	2431.12	2710	2634.78	1.02855	76.2836	189.593

While no known binding sites were identified during analysis of Tables 6.33, 6.34 and 6.35, the tables do provide some intuition into the makeup of the intronic regions. All three tables include words with strings of T's, indicating a high A/T content in the introns of *Arabidopsis*, which is verified by Table A.128.

6.2.2.8 The *Arabidopsis thaliana* Genome

Analysis of the full genome of *Arabidopsis* was performed using an input file consisting of the five chromosomes, each represented as a single sequence. The search for over-represented words in this file was performed using the $O * ln(\frac{O}{E})$ score due to the low number of sequences present and the extremely long average sequence length. Table 6.36 shows the timing and size data for the full genome tests and Tables 6.37, 6.38 and 6.39 show the top five words for lengths 6, 8 and 10.

Table 6.36: Size and Timing Information, Genome

Length	**Order**	**Size** (GB)	**Time (s)**			
			Enumerate	**Score**	**Cluster**	**Total**
6	4	0.000186	101.382673	13.816576	14.396031	115.199249
8	6	0.002971	156.262442	15.028098	0.471711	171.29054
10	8	0.047517	274.062791	32.666027	7.754121	306.728818

Although the full genome database for length 10 is slightly larger than the length 10 storage for the proximal promoters, the runtime is significantly lower, due to the minimal number of unique sequences in the full genome.

Table 6.37: Top 5 Words by $O * ln(\frac{O}{E})$ Score, Length 6, Genome

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAA	5	5	376532	321889	1.16976	59038.4	0
TTTTTT	5	5	372939	319693	1.16655	57453	0
TATATA	5	5	179209	145199	1.23423	37714	0
ATATAT	5	5	193956	171570	1.13048	23786.6	0
TAAAAT	5	5	128179	111987	1.14458	17309.4	0

Table 6.38: Top 5 Words by $O * ln(\frac{O}{E})$ Score, Length 8, Genome

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAAAA	5	5	128647	119326	1.07811	9676.06	0
TTTTTTTT	5	5	126568	117334	1.0787	9588.35	0
TATATATA	5	5	58214	49387.5	1.17872	9572.01	0
ATATATAT	5	5	59429	53451.6	1.11183	6299.82	0
TAAAAAAT	5	5	14818	11271.2	1.31467	4054.04	0

Table 6.39: Top 5 Words by $O * ln(\frac{O}{E})$ Score, Length 10, Genome

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATATATATA	5	5	31261	27617.1	1.13194	3874.33	0
ATATATATAT	5	5	31669	28193.5	1.12327	3681.38	0
CTTCTTCTTC	5	5	10149	8380.55	1.21102	1943.14	0
GAAGAAGAAG	5	5	9989	8396.54	1.18966	1734.73	0
ATTTTTTTTA	5	5	3285	2222.47	1.47808	1283.6	0

Tables 6.37, 6.38 and 6.39 show the top five over-represented words for lengths 6, 8 and 10 in the full genome of *Arabidopsis thaliana* and, although none of the words listed match to known binding sites in AGRIS, two interesting observations can be made. First, the high A/T content of the genome can be seen in the extremely high A/T content of the tables. Second, and possibly more interesting, is the fact that a poly-A word is the highest ranked word for both lengths 6 and 8 (Tables 6.37 and 6.38), but the extension of this word to length 10 does not appear in the length 10 table (Table 6.39), indicating that strings of A's are very likely to be less than 10 nucleotides in length across the entire genomic landscape.

7 Conclusions and Future Work

7.1 Closing Thoughts

While the development of the OWEF framework was driven by its contributions to the WordSeeker tool, a larger goal of easier development for bioinformatics word and motif discovery tools exists. Redundancy of code and functionality brings down the quality of tools and takes the focus away from new research. The OWEF framework can help developers focus on new algorithms and optimizations to existing algorithms, while keeping many common aspects consistent between workflows. The use of the framework as a basis for new tools can facilitate rapid implementation by removing much of the low-level needs such as input parsing and communication between stages.

To achieve widespread acceptance, the OWEF framework needs to be more flexible. This flexibility will be realized as developers add new child classes for the various stages of the framework. Three data structures are currently in place or being developed, each with its own strengths and weaknesses. Many more are possible, including hash tables, simple arrays and burst tries [26]. Similarly, the word scoring and clustering stages only contain one or two methods for performing their work, but many other approaches have been identified and their inclusion could bring the necessary flexibility to the framework.

The applications of the OWEF framework discussed in this document show another form of flexibility. The framework can be used to improve existing tools, in addition to being used to build new tools. The OpenMotif project is essentially a joint effort between the WordSeeker group and the R'MES group that takes the best algorithms from both groups and combines them to form a valuable, powerful "new" tool capable of providing detailed statistics about words contained in datasets of all sizes.

The need for this improvement can be seen in tables reported in Section 6.2.2, as the statistics used by WordSeeker appear somewhat insufficient for dealing with genome wide

input sets. While the tool is capable of processing the input, and the statistical analysis has been shown to be useful in [35], the quality of the results could possibly be improved with a more robust statistics package. Further analysis of the words reported in the segment analyses could also be performed using the TRANSFAC [69] and JASPAR [52] databases to identify words that match known binding sites in other organisms.

7.2 Direction for the Future

Several optimizations have been mentioned earlier in the document, including the removal of string variables from the framework, the addition of multi-level prefixes to allow processing on an arbitrary number of nodes and combining shared and distributed memory parallelism to increase processing speed. The implementation of these three major improvements will provide valuable gains in both processing speed and memory distribution, allowing the framework to process larger datasets, for longer word lengths, in less time.

An improvement that could prove incredibly useful for developers, but will require many modifications to the existing framework and classes, is the addition of parallel-aware interfaces. These interfaces would be implemented in each parent class, allowing the child processes to be written from a purely sequential viewpoint. For example, the MarkovModel class contains many sections of code that must be aware of whether the system is operating in shared or distributed memory space, and call the appropriate functions accordingly. The addition of parallel-aware interfaces would lay this task upon the framework, easing the incorporation of new algorithms.

Other improvements are currently underway, such as the creation of a new data structure, the RadixTrieGeneral class, which provides a radix trie capable of processing English language text or protein data, and is not limited to the basic DNA alphabet. Also currently underway is the full implementation of the three shell classes, bringing sequence

clustering and high-level genomic analysis into the framework. Like the other additional components, these stages may not be applicable in certain tools, and are only enabled when explicitly requested via the input parameters. Since the framework was developed and used concurrently, some sections were developed in the fastest way possible, and many other optimizations may be possible. As these inefficient segments are identified, they will be addressed and improved.

References

[1] Gnuplot Homepage, 2009.

[2] J. M. Alonso, A. N. Stepanova, T. J. Leisse, C. J. Kim, H. Chen, P. Shinn, D. K. Stevenson, J. Zimmerman, P. Barajas, R. Cheuk, C. Gadrinab, C. Heller, A. Jeske, E. Koesema, C. C. Meyers, H. Parker, L. Prednis, Y. Ansari, N. Choy, H. Deen, M. Geralt, N. Hazari, E. Hom, M. Karnes, C. Mulholland, R. Ndubaku, I. Schmidt, P. Guzman, L. Aguilar-Henonin, M. Schmid, D. Weigel, D. E. Carter, T. Marchand, E. Risseeuw, D. Brogden, A. Zeko, W. L. Crosby, C. C. Berry, and J. R. Ecker. Genome-wide insertional mutagenesis of Arabidopsis thaliana. *Science (New York, N.Y.)*, 301(5633):653–7, August 2003.

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–10, October 1990.

[4] A. Apostolico, F. Gong, and S. Lonardi. Verbumculus and the Discovery of Unusual Words. *Journal of Computer Science and Technology*, 19(1):22–41, 2004.

[5] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, and Others. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):2529, 2000.

[6] A. D. Birrell and B. J. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, February 1984.

[7] M. Blanchette and S. Sinha. Separating real motifs from their artifacts. *Bioinformatics (Oxford, England)*, 17 Suppl 1:S30–8, January 2001.

[8] P. Bornstein, J. McKay, J. K. Morishima, S. Devarayalu, and R. E. Gelinas. Regulatory elements in the first intron contribute to transcriptional control of the human alpha 1(I) collagen gene. *Proceedings of the National Academy of Sciences of the United States of America*, 84(24):8869–73, December 1987.

[9] D. R. Butenhof. *Programming with POSIX threads*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[10] D. Callahan and K. Kennedy. Compiling programs for distributed-memory multiprocessors. *The Journal of Supercomputing*, 2(2):151–169, October 1988.

[11] P. J. a. Cock, T. Antao, J. T. Chang, B. a. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics (Oxford, England)*, 25(11):1422–3, June 2009.

[12] A. Cornish-Bowden. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations. *Nucleic acids research*, 13(9):3021–3030, 1986.

[13] L. Dagum and R. Menon. OpenMP : An Industry-Standard API for Shared-Memory Programming. *IEEE Computational Science \& Engineering*, 5(1):4655, 1998.

[14] M. K. Das and H.-K. Dai. A survey of DNA motif finding algorithms. *BMC bioinformatics*, 8 Suppl 7:S21, January 2007.

[15] R. V. Davuluri, H. Sun, S. K. Palaniswamy, N. Matthews, C. Molina, M. Kurtz, and E. Grotewold. AGRIS : Arabidopsis Gene Regulatory Information Server , an transcription factors. *BMC Bioinformatics*, 11:1–11, 2003.

[16] F. Drews, J. Lichtenberg, and L. Welch. Scalable parallel word search in multicore/multiprocessor systems. *The Journal of Supercomputing*, 51(1):58–75, 2009.

[17] J. Dutheil, S. Gaillard, E. Bazin, S. Glémin, V. Ranwez, N. Galtier, and K. Belkhir. Bio++: a set of C++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics. *BMC bioinformatics*, 7:188, January 2006.

[18] E. Eskin and P. a. Pevzner. Finding composite regulatory patterns in DNA sequences. *Bioinformatics (Oxford, England)*, 18 Suppl 1(1):S354–63, January 2002.

[19] F. Fauteux, M. Blanchette, and M. V. Strömvik. Seeder: discriminative seeding DNA motif discovery. *Bioinformatics (Oxford, England)*, 24(20):2303–7, October 2008.

[20] G. Giuliano, E. Pichersky, V. S. Malik, M. P. Timko, P. a. Scolnik, and a. R. Cashmore. An evolutionarily conserved protein binding sequence upstream of a plant light-regulated gene. *Proceedings of the National Academy of Sciences of the United States of America*, 85(19):7089–93, October 1988.

[21] N. Goto. BioRuby : Open-Source Bioinformatics Library. *Information Research*, 630:629–630, 2003.

[22] S. Graham, P. Kessler, and M. Mckusick. Gprof: A call graph execution profiler. *ACM Sigplan Notices*, 17(6):126, 1982.

[23] W. Gropp. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, September 1996.

[24] R. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29(2):147160, 1950.

[25] R. A. Hankins and J. M. Patel. Practical Suffix Tree Construction. *Very Large Data Bases*, 30:36–47, 2004.

[26] S. Heinz, J. Zobel, and H. E. Williams. Burst tries: a fast, efficient data structure for string keys. *ACM Trans. Inf. Syst.*, 20(2):192–223, 2002.

[27] M. Hoebeke and S. Schbath. R'MES: Finding Exceptional Motifs, 2006.

[28] R. C. G. Holland, T. a. Down, M. Pocock, a. Prlić, D. Huen, K. James, S. Foisy, a. Dräger, a. Yates, M. Heuer, and M. J. Schreiber. BioJava: an open-source framework for bioinformatics. *Bioinformatics (Oxford, England)*, 24(18):2096–7, September 2008.

[29] R. Hong, L. Hamaguchi, M. Busch, and D. Weigel. Regulatory elements of the floral homeotic gene AGAMOUS identified by phylogenetic footprinting and shadowing. *The Plant Cell Online*, 15(6):1296, 2003.

[30] T. Kohonen. Self-organizing maps of symbol strings. *Neurocomputing*, 21(1-3):19–30, November 1998.

[31] S. Kurtz and C. Schleiermacher. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics (Oxford, England)*, 15(5):426–7, May 1999.

[32] C. Larman and V. Basili. Iterative and incremental development: a brief history. *Computer*, 36(6):47–56, June 2003.

[33] E. a. Leibold and H. N. Munro. Cytoplasmic protein binds in vitro to a highly conserved sequence in the 5' untranslated region of ferritin heavy- and light-subunit mRNAs. *Proceedings of the National Academy of Sciences of the United States of America*, 85(7):2171–5, April 1988.

[34] X. Liang, K. Shen, J. Lichtenberg, S. E. Wyatt, and L. R. Welch. An integrated bioinformatics approach to the discovery of CIS-regulatory elements involved in plant gravitropic signal transduction. *International Journal of Computational Bioscience*, 1(1):33–54, 2010.

[35] J. Lichtenberg, E. Jacox, J. D. Welch, K. Kurz, X. Liang, M. Q. Yang, F. Drews, K. Ecker, S. S. Lee, L. Elnitski, and L. R. Welch. Word-based characterization of promoters involved in human DNA repair pathways. *BMC genomics*, 10 Suppl 1:S18, 2009.

[36] J. Lichtenberg, A. Yilmaz, J. D. Welch, K. Kurz, X. Liang, F. Drews, K. Ecker, S. S. Lee, M. Geisler, E. Grotewold, and L. R. Welch. The word landscape of the non-coding segments of the Arabidopsis thaliana genome. *BMC genomics*, 10:463, 2009.

[37] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, page 319327. Society for Industrial and Applied Mathematics, 1990.

[38] E. M. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *Journal of the ACM*, 23(2):262–272, April 1976.

[39] D. W. Meinke, J. M. Cherry, C. Dean, S. D. Rounsley, and M. Koornneef. Arabidopsis thaliana: a model plant for genome analysis. *Science (New York, N.Y.)*, 282(5389):662, 679–82, October 1998.

[40] K. Morohashi, M. Zhao, M. Yang, B. Read, A. Lloyd, R. Lamb, and E. Grotewold. Participation of the Arabidopsis bHLH factor GL3 in trichome initiation regulatory events. *Plant physiology*, 145(3):736–46, November 2007.

[41] S. K. Palaniswamy, S. James, H. Sun, R. S. Lamb, R. V. Davuluri, and E. Grotewold. AGRIS and AtRegNet. A Platform to Link cis-Regulatory Elements and Transcription Factors into Regulatory Networks. *Society*, 140(March):818–829, 2006.

[42] G. Pavesi, G. Mauri, and G. Pesole. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics (Oxford, England)*, 17 Suppl 1:S207–14, January 2001.

[43] G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole. Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic acids research*, 32(Web Server issue):W199–203, 2004.

[44] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85(8):2444–8, April 1988.

[45] P. a. Pevzner and S. H. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *Proceedings / ... International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, 8:269–78, January 2000.

[46] S. Rajasekaran, S. Balla, C. Huang, V. Thapar, M. Gryk, M. Maciejewski, and M. Schiller. High-performance exact algorithms for motif search. *Journal of Clinical Monitoring and Computing*, 19(4):319–328, 2005.

[47] L. H. Reid, R. G. Gregg, O. Smithies, and B. H. Koller. Regulatory elements in the introns of the human HPRT gene are necessary for its expression in embryonic stem cells. *Proceedings of the National Academy of Sciences of the United States of America*, 87(11):4299–303, June 1990.

[48] G. Reinert, S. Schbath, and M. S. Waterman. Probabilistic and statistical properties of words: an overview. *Journal of computational biology : a journal of computational molecular cell biology*, 7(1-2):1–46, 2000.

[49] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. *Bioinformatics-Oxford*, 14(1):5567, 1998.

[50] E. Ristad and P. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, May 1998.

[51] M.-f. Sagot. Spelling approximate repeated or common motifs using a suffix tree. *Latin'98: Theoretical Informatics*, page 374390, 1998.

[52] A. Sandelin, W. Alkema, P. Engström, W. W. Wasserman, and B. Lenhard. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic acids research*, 32(Database issue):D91–4, 2004.

[53] S. Schbath, B. Prum, and E. De Turckheim. Exceptional motifs in different Markov chain models for a statistical analysis of DNA sequences. *Journal of Computational Biology*, 2(3):417437, 1995.

[54] Y. Shen, a. Danon, and D. a. Christopher. RNA binding-proteins interact specifically with the Arabidopsis chloroplast psbA mRNA 5' untranslated region in a redox-dependent manner. *Plant & cell physiology*, 42(10):1071–8, October 2001.

[55] S. Sinha. YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 31(13):3586–3588, 2003.

[56] S. Sinha and M. Tompa. A statistical method for finding transcription factor binding sites. *Proc Int Conf Intell Syst Mol Biol*, 2000.

[57] S. Sinha and M. Tompa. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Research*, 30(24):5549–5560, 2002.

[58] S. T. Smale and J. T. Kadonaga. The RNA polymerase II core promoter. *Annual review of biochemistry*, 72:449–79, January 2003.

[59] A. Smit, R. Hubley, and P. Green. RepeatMasker Open-3.0, 1996.

[60] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. a. Chervitz, C. Dagdigian, G. Fuellen, J. G. R. Gilbert, I. Korf, H. Lapp, H. Lehväslaiho, C. Matsalla, C. J. Mungall, B. I. Osborne, M. R. Pocock, P. Schattner, M. Senger, L. D. Stein, E. Stupka, M. D. Wilkinson, and E. Birney. The Bioperl toolkit: Perl modules for the life sciences. *Genome research*, 12(10):1611–8, October 2002.

[61] D. Swarbreck, C. Wilks, P. Lamesch, T. Z. Berardini, M. Garcia-Hernandez, H. Foerster, D. Li, T. Meyer, R. Muller, L. Ploetz, A. Radenbaugh, S. Singh, V. Swing, C. Tissier, P. Zhang, and E. Huala. The Arabidopsis Information Resource (TAIR): gene structure and function annotation. *Nucleic acids research*, 36(Database issue):D1009–14, 2008.

[62] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. a. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Régnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden,

M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nature biotechnology*, 23(1):137–44, January 2005.

[63] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, September 1995.

[64] J. Venter, M. Adams, E. Myers, P. Li, R. Mural, G. Sutton, H. Smith, M. Yandell, C. Evans, R. Holt, and Others. The sequence of the human genome. *Science*, 291(5507):1304, 2001.

[65] G. Wang, T. Yu, and W. Zhang. WordSpy: identifying transcription factor binding motifs by building a dictionary and learning a grammar. *Nucleic acids research*, 33(Web Server issue):W412–6, July 2005.

[66] M. S. Waterman. Pattern recognition in several sequences: consensus and alignment. *Bulletin of Mathematical Biology*, 46(4):515–527, 1984.

[67] P. Weiner. Linear Pattern Matching Algorithms. *14th Annual Symposium on Switching & Automata Theory*, pages 1–11, 1973.

[68] M. Wickens, D. S. Bernstein, J. Kimble, and R. Parker. A PUF family portrait: 3'UTR regulation as a way of life. *Trends in genetics : TIG*, 18(3):150–7, March 2002.

[69] E. Wingender, X. Chen, R. Hehl, H. Karas, I. Liebich, V. Matys, T. Meinhardt, and Others. TRANSFAC: an integrated system for gene expression regulation. *Nucleic acids research*, 28(1):316, 2000.

[70] S. Wyatt, A. Rashotte, M. Shipp, D. Robertson, and G. Muday. Mutations in the Gravity Persistence Signal Loci in Arabidopsis Disrupt the Perception and / or Signal Transduction of Gravitropic Stimuli 1. *Plant physiology*, 130(3):1426, 2002.

[71] A. Yilmaz, K. Kurz, and E. Grotewold. Arabidopsis thaliana word query, 2009.

Appendix: The Word Landscape of *Arabidopsis thaliana*

The following sections contain the runtimes for all word lengths, 2 to 20, and the top 25 words sorted by the $S * ln(\frac{S}{E_s})$ score for the various segments of the *Arabidopsis thaliana* genome and $O * ln(\frac{O}{E})$ for the full genome.

A.1 The *Arabidopsis thaliana* 5' Untranslated Regions

A.1.1 Timing Information

Table A.1: Full Size and Timing Information, 5' UTR

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	0.327075	0.670373	4.641517	0.997448
2	0	0.000001	0.585621	0.895685	8.451222	1.481306
3	1	0.000003	0.899247	0.794867	4.552653	1.694114
4	2	0.000012	1.357128	0.546236	1.739703	1.903364
5	3	0.000046	1.847039	0.837291	1.130908	2.68433
6	4	0.000186	2.299695	2.917490	1.121598	5.217185
7	5	0.000743	2.910841	5.890592	1.023159	8.801433
8	6	0.002958	3.590521	19.253780	1.416932	22.844301
9	7	0.011160	4.738977	68.353866	2.653307	73.092843
10	8	0.035570	6.154461	157.014079	5.855332	163.16854
11	9	0.089114	7.473788	312.312749	10.895260	319.786537
12	10	0.176771	9.115835	391.781266	16.474382	400.897101
13	11	0.291711	10.511402	493.489195	21.224632	504.000597
14	12	0.423010	12.150961	538.677117	25.375064	550.828078
15	13	0.562426	13.700427	553.863636	28.788339	567.564063
16	14	0.705372	15.211259	553.973944	31.705563	569.185203
17	15	0.849574	16.970070	449.145713	36.506009	466.115783
18	16	0.993926	18.725925	587.280915	37.273575	606.00684
19	17	1.137867	20.390068	487.129606	40.520588	507.519674
20	18	1.281115	22.213683	567.362809	43.134395	589.576492

A.1.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.2: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
G	18734	18847.6	431987	431987	1	0	-113.278
C	18836	18923.1	563336	563336	1	0	-86.9054
T	18775	18998.6	853525	853525	1	0	-222.299
A	19021	18985.3	779200	779200	1	0	35.6871

Table A.3: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AC	17949	18113.2	126325	167026	0.756321	-35281.2	-163.455
CA	18226	18113.2	157588	167026	0.943496	-9165.78	113.149
AG	17837	17705.3	154680	128081	1.20767	29186.9	132.163
CG	15283	16997	79516	92598.7	0.858716	-12111.6	-1624.55
AT	18070	18523.7	186039	253065	0.735144	-57242.2	-448.059
GC	15642	16997	69694	92598.7	0.752646	-19804.3	-1299.52
CC	16306	17595.2	103375	120754	0.856079	-16063.7	-1240.8
GT	16502	17860.9	114920	140299	0.81911	-22930.8	-1305.82
AA	18576	18452.5	303635	231028	1.31428	82979.8	123.931
GA	17852	17705.3	171255	128081	1.33708	49747.4	147.28
CT	17662	18223.9	219225	182958	1.19823	39645.6	-553.163
GG	13844	16215.8	71648	71008.1	1.00901	642.774	-2189.18
TG	16261	17860.9	121895	140299	0.868825	-17140.1	-1525.99
TA	17558	18523.7	137178	253065	0.542067	-84003.2	-940.04
TC	17991	18223.9	260963	182958	1.42636	92674.4	-231.422
TT	17792	18586.4	331024	277204	1.19415	58736.3	-777.166

Table A.4: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ACC	11860	11030.9	26730	23181.3	1.15309	3807.47	859.466
ACA	14229	13215	45271	35338.2	1.28108	11213.7	1051.97
AAC	15516	14744.8	49034	49225.7	0.996105	-191.359	791.011
ACG	9075	9628.68	17415	17831	0.976668	-411.134	-537.445
AAG	15374	15556.9	59128	60275	0.980971	-1135.99	-181.846
AGG	9384	11570.1	20580	25654.7	0.802191	-4536.01	-1965.19
ACT	13550	14739.1	36088	49160	0.734093	-11155.5	-1139.81
AGT	12270	13945.1	31158	41149	0.7572	-8665.92	-1570.21
ATG	6580	11755	16327	26568.9	0.614515	-7949.96	-3817.99
AAT	15621	16204.1	62414	72494.8	0.860944	-9344.95	-572.466
AGA	15568	15621.1	74235	61320.6	1.2106	14187.7	-52.976
ATC	15508	15335.1	59117	56880.9	1.03931	2279.46	173.871
CAG	11324	12603.1	24342	31283	0.778122	-6106.71	-1211.87
CAA	16111	15626.3	60864	61408.2	0.991139	-541.734	492.09
CAT	13289	13520.8	38126	37625.1	1.01331	504.173	-229.782
CCA	12965	12198.3	33039	28918.2	1.1425	4401.38	790.314
ATA	12776	12371.1	39910	29900.1	1.33478	11524.7	411.456
CAC	12286	11548.1	31712	25548.4	1.24125	6853.63	760.957
CCC	8572	9958.45	18827	18969.8	0.99247	-142.295	-1285.12
CCG	8264	8577.13	17290	14591.6	1.18493	2933.82	-307.346
CGC	7343	7922.26	13290	12828.6	1.03597	469.601	-557.55
CGG	7363	8061.2	14196	13188.3	1.07641	1045.28	-667.048
CCT	11974	13839.4	32792	40228.9	0.815136	-6702.71	-1733.59
CGT	9845	10541.3	22195	21153.4	1.04924	1066.87	-672.733
AGC	11760	11423.6	26220	24955.1	1.05069	1296.46	341.339

Table A.5: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAC	11793	10396.1	23615	20854.7	1.13236	2935.47	1486.78
GACG	3517	2415.22	4808	2798.67	1.71796	2601.79	1321.76
CGCC	3209	2185.8	4304	2501.67	1.72045	2335.29	1232.18
ATCA	9291	8161.58	17109	13603.6	1.25768	3922.59	1204.19
ACTA	4980	3910.53	6728	4927.62	1.36536	2095.24	1203.95
GGCG	2446	1528.34	3102	1689.27	1.83629	1885.24	1150.29
CAAA	12634	11543.3	27939	25886.1	1.07931	2132.27	1140.7
CGTC	4882	3874.49	7409	4872.01	1.52073	3105.77	1128.43
GAAG	8637	7622.52	18576	12208.3	1.52159	7797.43	1079.18
CCTA	4531	3614.52	5806	4477.57	1.29668	1508.46	1023.93
CGAC	3757	2868.3	5042	3407.15	1.47983	1976.1	1014.03
AACC	7784	6844.21	11897	10375.5	1.14665	1628.03	1001.55
GTCG	3621	2760.84	4981	3260.13	1.52786	2111.27	982.074
GCCG	2757	1942.28	3719	2194.22	1.69491	1962.25	965.717
AATC	10906	10128.7	21810	19833.1	1.09968	2072.32	806.432
CGGC	2611	1924.25	3407	2171.76	1.56877	1534.15	796.871
AAGC	7430	6684.33	11165	10022.9	1.11395	1204.87	785.795
GCAG	3126	2444.13	3991	2836.62	1.40696	1362.65	769.205
TAAT	7268	6546.74	12384	9725.47	1.27336	2992.68	759.607
ACCA	6625	5973.36	9850	8543	1.15299	1402.24	685.953
ACCC	4503	3871.99	5892	4868.16	1.21031	1124.67	679.846
ATTA	7322	6681.46	12320	10016.6	1.22996	2549.99	670.304
CTAA	7414	6821.23	10928	10324.3	1.05847	621.012	617.808
ATCG	6060	5505.01	9069	7640.09	1.18703	1554.91	582.075
ACGA	5302	4752.22	7470	6294.86	1.18668	1278.58	580.426

Table A.6: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CCCTA	1994	1122.16	2221	1223.98	1.81457	1323.38	1146.33
AACCC	3091	2262.96	3641	2622.41	1.38842	1194.84	963.843
ACTCA	2616	1852.54	3009	2099.9	1.43292	1082.39	902.752
TAGGG	1573	895.757	1801	965.645	1.86507	1122.56	885.711
ATTCA	3963	3280.21	4980	4021.93	1.23821	1064.07	749.378
AAACC	5074	4382.23	6576	5729.65	1.14771	905.99	743.711
TCAAT	4169	3501.23	5419	4347.39	1.24649	1193.99	727.746
TGAAT	2938	2298.24	3718	2668.41	1.39334	1233.27	721.529
CCTAA	2481	1868.34	2801	2119.6	1.32148	780.782	703.644
ACTTG	2070	1505.08	2383	1674.89	1.42278	840.278	659.719
CGCCG	1638	1118.8	2087	1220.11	1.71051	1120.28	624.44
ATCCA	2850	2297.74	3344	2667.76	1.25348	755.501	613.864
CAGAT	2644	2107.33	3146	2421.6	1.29914	823.321	599.843
TAGAT	2252	1733.11	2710	1952.07	1.38827	889.044	589.803
CACCG	1378	900.438	1548	970.926	1.59435	722.094	586.35
CAAGT	2137	1632.91	2412	1829.46	1.31842	666.766	574.937
CGGCG	1363	894.469	1648	964.193	1.7092	883.372	574.114
ACTTT	4229	3697.69	5367	4643.59	1.15579	777.036	567.776
CCATT	3074	2572.48	3657	3031.94	1.20616	685.474	547.503
ACATA	2115	1652.89	2527	1853.8	1.36314	782.846	521.408
ATCTA	2591	2121.08	3040	2439.21	1.24631	669.359	518.501
AGAAA	7619	7119.31	11825	11118.2	1.06357	728.804	516.83
CAAAT	4587	4118.5	5850	5300.97	1.10357	576.533	494.187
ATTCG	2429	1983.52	2843	2264.19	1.25564	647.19	492.132
CGAAT	2062	1627.16	2349	1822.46	1.28891	596.176	488.36

Table A.7: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTCTTT	3608	2979.93	4622	3623.72	1.27549	1124.66	690.041
ACCCTA	1294	763.683	1381	824.08	1.67581	713.004	682.379
TTCTCC	2910	2302.65	3391	2696.69	1.25747	776.874	681.214
CAAAAC	2316	1869.71	2643	2138.81	1.23573	559.426	495.753
TAGGGT	1155	777.041	1268	839.075	1.51119	523.552	457.798
AAACCC	2173	1768.89	2457	2012.54	1.22084	490.276	447.115
AAAGAG	2282	1879.49	2666	2151.13	1.23935	572.085	442.823
ATTTTC	2688	2294.76	3155	2686.29	1.17448	507.408	425.163
TTTCTC	4878	4477.64	6621	5943.33	1.11402	714.91	417.751
CAAAAT	2153	1774.85	2371	2019.98	1.17377	379.893	415.84
TGAAAT	1200	852.653	1316	924.341	1.42372	464.905	410.07
ACAGAG	1395	1042.28	1558	1141.13	1.36531	485.136	406.616
ATTCTC	2354	1981.92	2635	2280.93	1.15523	380.226	405.003
ATCTCA	1625	1269.2	1770	1406.21	1.25871	407.248	401.566
CATCTT	1807	1448.93	1988	1620.65	1.22667	406.154	399.058
ACTTCA	1030	702.283	1093	755.421	1.44688	403.761	394.466
CTCTGT	1995	1652.06	2308	1867.92	1.2356	488.276	376.298
TGAGAT	1005	696.024	1088	748.445	1.45368	407.019	369.196
ACAAAA	3545	3197.24	4352	3936.15	1.10565	437.078	366.023
AACCCT	1739	1419.34	1910	1585.06	1.205	356.18	353.225
TCAAAT	1877	1557.31	2089	1751.92	1.1924	367.605	350.46
TCAGAT	1341	1035.36	1477	1133.14	1.30346	391.437	346.873
TCCCTC	1008	722.333	1108	777.794	1.42454	392.066	335.903
GGAGAA	1365	1074.28	1476	1178.13	1.25283	332.696	326.919
TCTTTC	3762	3453.54	5014	4314.6	1.1621	753.251	321.84

Table A.8: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAC	1083	683.968	1132	740.797	1.52808	479.985	497.725
AAAGAAA	1816	1449.57	2193	1634.64	1.34158	644.412	409.268
TTTCTTT	1909	1544.86	2366	1750.99	1.35123	712.208	404.029
CTTCTCC	1327	1038.13	1433	1145.42	1.25107	320.987	325.773
CATCTTC	1037	766.358	1105	833.603	1.32557	311.437	313.629
CTCTTTC	1753	1472.45	2037	1662.46	1.22529	413.876	305.73
GTTCTTG	377	168.841	392	178.066	2.20143	309.329	302.84
TTCTCTT	2084	1806.9	2482	2077.08	1.19494	442.044	297.338
CTCTTCC	893	651.358	939	704.281	1.33327	270.092	281.766
CTTTCTC	1875	1616.05	2157	1838.68	1.17312	344.405	278.672
GTTTTTG	775	546.694	844	587.911	1.43559	305.171	270.454
GAAAAAG	811	581.123	847	626.053	1.35292	256.02	270.31
TAAAAAT	616	399.002	664	425.819	1.55935	294.994	267.516
CTTTTTC	1148	911.002	1233	998.463	1.2349	260.148	265.453
CGTCTTC	709	492.087	754	527.692	1.42886	269.087	258.926
AACCCTA	1169	936.991	1244	1028.35	1.20971	236.832	258.619
TCAAAAT	790	572.19	815	616.143	1.32275	227.963	254.824
AAGAGAA	1132	906.223	1270	992.977	1.27898	312.503	251.82
CAAGAAC	351	184.258	366	194.48	1.88194	231.423	226.201
CATTTTC	838	640.475	884	692.123	1.27723	216.31	225.262
GAGAAAG	934	734.241	1001	797.332	1.25544	227.711	224.757
ACAAAAA	1737	1526.75	1921	1728.79	1.11118	202.518	224.099
ATAAATA	552	370.935	594	395.292	1.50268	241.909	219.431
CTCTCAC	791	599.765	818	646.76	1.26477	192.137	218.918
TTCTCTG	1074	875.985	1138	958.325	1.18749	195.555	218.878

Table A.9: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTCTTCTC	877	620.404	999	675.022	1.47995	391.618	303.561
CTTTCTCT	1173	1021.67	1314	1135.35	1.15735	192.023	162.024
AACAAAAA	1059	929.997	1142	1028.48	1.11038	119.57	137.563
TTTCTTCA	617	497.591	637	537.934	1.18416	107.675	132.711
GAGAAGAG	320	212.597	365	226.456	1.61179	174.232	130.856
TTCTCTCC	461	351.303	470	376.905	1.247	103.748	125.277
CTTTCTTC	894	780.719	941	856.623	1.0985	88.4021	121.129
TTTCTCTC	1446	1330.75	1581	1503.37	1.05164	79.6034	120.107
CTCTCTTT	1244	1130.74	1368	1263.85	1.08241	108.326	118.754
AGAAAAAA	1089	979.544	1165	1086.11	1.07263	81.687	115.356
AAAGAGAG	672	567.735	716	616.016	1.16231	107.691	113.302
AAAGAAAA	985	885.093	1100	976.499	1.12647	131	105.345
AAAAAACA	766	667.76	810	728.35	1.1121	86.0641	105.136
AAGAAAAA	1119	1019.35	1222	1132.63	1.07891	92.8087	104.374
TTCTCTCA	489	396.718	500	426.635	1.17196	79.3397	102.268
ATCTCTCA	334	247.681	344	264.307	1.30152	90.6547	99.8657
TCTTCTCC	921	827.056	960	909.681	1.05531	51.6853	99.0883
AGAGAAAG	592	501.562	638	542.339	1.17639	103.641	98.1413
ACAAAAAA	853	761.02	909	834.145	1.08974	78.1175	97.3274
TTTTTGTT	824	732.509	899	801.693	1.12138	102.987	96.9807
TTTTTCTT	1030	937.546	1128	1037.24	1.0875	94.6211	96.8692
CAAAAACC	359	275.182	364	294.072	1.23779	77.6513	95.4545
CCTCTCTT	353	270.074	360	288.538	1.24767	79.6604	94.5234
CTCCTCTC	409	325.488	420	348.741	1.20433	78.0892	93.4115
TAAAAAAG	232	155.282	239	164.916	1.44923	88.6762	93.1461

Table A.10: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTCTCTT	582	428.09	618	464.76	1.32972	176.11	178.754
CTCTTTCTC	668	531.343	745	579.994	1.2845	186.523	152.892
TTTCTTCTC	491	362.347	504	392.033	1.28561	126.621	149.186
AAGAGAGAA	307	194.15	317	208.218	1.52244	133.24	140.672
TTCTTCTCC	510	420.72	523	456.582	1.14547	71.0304	98.1458
CTTCTTCCT	555	465.807	573	506.709	1.13083	70.4502	97.2348
CTCTTCTTT	241	165.212	245	176.917	1.38483	79.7667	90.9931
CTCTCTTTC	671	589.271	718	645.19	1.11285	76.7721	87.1519
GAGAAAGAG	276	203.381	298	218.223	1.36558	92.8502	84.2688
TTCTCTCTG	254	182.46	257	195.562	1.31416	70.2124	84.0243
TTCTTTCTT	457	385.716	515	417.827	1.23257	107.686	77.4985
CTCTTCTCT	541	470.74	618	512.208	1.20654	116.034	75.2602
TCTCTCTTT	940	868.239	1013	964.767	1.04999	49.4195	74.6477
TTTCTCTCT	983	912.573	1056	1016.42	1.03894	40.3384	73.0772
AAAGAAGAG	149	91.3127	152	97.4064	1.56047	67.6383	72.9588
AAGAAAAAA	631	562.454	662	614.959	1.07649	48.796	72.5631
TTTCTTCTA	161	103.319	162	110.282	1.46896	62.2976	71.4175
TTTCTTCTG	161	104.008	166	111.022	1.49519	66.7744	70.3463
ATTCTTCTC	173	116.599	176	124.544	1.41316	60.8654	68.2572
CTTTCTCTC	609	545.693	635	596.107	1.06524	40.1347	66.8451
CTTCTTCGT	291	233.594	296	251.036	1.17912	48.7702	63.9444
TTTCTCTTT	428	370.344	464	400.853	1.15753	67.879	61.9272
AACAAAAAA	554	495.977	580	540.384	1.07331	41.0337	61.2916
AAAAAAAGA	481	424.046	500	460.272	1.08632	41.3956	60.6179
ACCAAAAAA	292	238.493	303	256.366	1.1819	50.6395	59.1054

Table A.11: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTCTTCTTT	192	108.694	192	116.959	1.64159	95.1682	109.24
CTCTCTCTTT	532	444.414	552	486.735	1.13409	69.4568	95.6995
AAAGAAGAAA	156	89.0452	157	95.7178	1.64024	77.6902	87.4711
CTTTCTCTCT	455	388.406	471	424.137	1.11049	49.362	72.0018
ACTCTCTCTT	115	66.3207	116	71.2056	1.62908	56.6101	63.2995
TTCTCTCTCA	141	93.5084	142	100.539	1.41239	49.03	57.91
CTTCTTCCTC	342	290.101	347	315.15	1.10106	33.4078	56.2873
CTCTTTCTCT	494	441.505	547	483.475	1.13139	67.5273	55.4991
TCTTTCTCTC	452	399.821	466	436.865	1.06669	30.0861	55.4449
AAAAAAAAAC	473	421.304	490	460.861	1.06323	30.0418	54.7457
TCTTCTTCCT	455	405.761	464	443.495	1.04624	20.9722	52.1121
CTCTCTTCTC	317	270.494	334	293.547	1.13781	43.1209	50.2932
TCTTCTCTCT	463	415.658	478	454.549	1.05159	24.0458	49.9413
AAAAAAAAAG	418	371.577	434	405.399	1.07055	29.5871	49.2093
TTCTCTCTTC	256	212.424	268	229.825	1.1661	41.1831	47.7679
CTCTTCTTCA	136	96.6999	137	103.988	1.31746	37.7719	46.3818
CATCTTCTTC	268	226.786	272	245.549	1.10772	27.8274	44.7508
CTTCTCTCTC	380	337.901	387	368.004	1.05162	19.4785	44.6188
GTTTTTTTTT	258	217.152	267	234.999	1.13618	34.0872	44.4692
CCTCTCTCTT	104	69.8578	106	75.0171	1.41301	36.6466	41.3847
CTCTCTTTCT	461	421.748	492	461.357	1.06642	31.6384	41.0244
AAAGAAAAAA	346	307.403	358	334.25	1.07105	24.574	40.925
CAAAAAAAAA	537	498.924	573	548.015	1.04559	25.5464	39.493
AAAAAAAATC	196	160.492	198	173.166	1.14341	26.5356	39.1751
AAAAAGAAAG	119	85.8127	119	92.2274	1.29029	30.3291	38.9079

Table A.12: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTCTCTCTA	106	57.7526	109	62.4655	1.74496	60.6839	64.3707
CTCTCTCTTTC	327	272.468	335	298.069	1.1239	39.1297	59.6581
TCTCTCTCTTT	408	361.183	421	396.986	1.06049	24.7258	49.7283
TCTTCTTCCTC	297	257.066	301	280.991	1.07121	20.7048	42.8861
ATCTTCTTCTC	97	63.4587	97	68.658	1.4128	33.5206	41.1591
ATCTCTCTCTT	114	80.1597	114	86.8038	1.31331	31.0705	40.1482
ATCTCTCTCTG	62	32.8001	62	35.4301	1.74992	34.6935	39.4755
TTTCTTCTTCC	104	72.4651	105	78.4395	1.33861	30.6215	37.5738
AAGAAAAAAAA	254	220.278	259	240.31	1.07777	19.3983	36.1804
CTTTCTTCTTC	213	180.749	218	196.774	1.10787	22.3313	34.9705
CTCTTCTTCTC	107	77.3043	109	83.699	1.30228	28.7891	34.7835
CTTTCTCTCTC	249	217.132	255	236.838	1.07669	18.8412	34.1003
TTTCTCTCTCT	365	333.711	377	366.256	1.02933	10.8998	32.7118
AAAAAAAAAAC	322	292.089	328	319.867	1.02543	8.23587	31.3932
TTCTCTCTTTT	90	63.5174	92	68.7216	1.33873	26.8387	31.3647
AAAAAAAAATC	140	112.329	141	121.846	1.1572	20.5861	30.8296
CTCTTTCTCTC	287	257.951	295	281.971	1.04621	13.3252	30.6267
TTCTTCTTCCT	301	272.251	305	297.829	1.02408	7.25644	30.2158
GTTTTTTTTTT	163	135.878	168	147.574	1.13841	21.7785	29.6648
TAGAGAGAGAA	53	30.3829	54	32.8148	1.6456	26.8976	29.4899
TTCTTCTCTTT	109	83.3153	109	90.2359	1.20794	20.5923	29.2901
GAAAGAAAAAA	102	76.6359	102	82.9724	1.22932	21.0594	29.1625
TCTTTCTCTCT	343	315.243	352	345.647	1.01838	6.41078	28.9451
GAAGAAGAAAG	97	73.757	97	79.8434	1.21488	18.8804	26.5716
CTTTTTTTTTT	164	139.919	169	151.996	1.11187	17.9218	26.0432

Table A.13: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTCTTCTTCTC	85	51.4734	85	56.096	1.51526	35.3249	42.6348
CTTCTCTCTCTC	183	154.707	183	169.529	1.07946	13.9922	30.736
CTCTCTCTCTTT	268	241.93	272	266.347	1.02122	5.71263	27.4268
CTTTTTTTTTTC	48	27.3909	48	29.8126	1.61006	22.861	26.9275
CTTCTTCTTCCT	220	195.482	223	214.678	1.03876	8.48114	25.995
CATCTCTCTCTC	99	77.7026	99	84.7989	1.16747	15.3288	23.9809
CAAAAAAAAAAA	249	226.202	258	248.822	1.03688	9.34478	23.9101
TCTTCTTCTTCC	268	245.413	271	270.232	1.00284	0.76898	23.5956
AAAAAAAAAACT	105	84.614	105	92.3755	1.13666	13.4503	22.6654
GAAAAAAAAAAT	35	18.4102	35	20.0283	1.74752	19.537	22.4854
CTCTCTTTCTCT	232	210.618	236	231.488	1.01949	4.55593	22.4319
CTCTTCTCTCTC	106	85.7862	106	93.6611	1.13174	13.1182	22.4276
CCAAAAAAAAAA	127	106.507	129	116.412	1.10813	13.2455	22.3495
GAAAAAAAAAAC	51	33.1529	51	36.095	1.41294	17.6292	21.9655
TCTCTCTCTTTC	252	232.196	256	255.498	1.00196	0.502304	20.625
TCTCTCTTTCTC	231	211.792	237	232.793	1.01807	4.24511	20.0534
CTTTCTCTCTCT	183	165.124	186	181.045	1.02737	5.02199	18.8108
AGGAAGAAGAAG	81	64.7164	81	70.578	1.14767	11.1561	18.1791
CTTCGTCTTCTT	72	55.9956	72	61.039	1.17957	11.8911	18.1003
TTTTTCTTTCTT	72	56.003	72	61.0471	1.17942	11.8814	18.0907
CTCTCTCTTTCT	228	210.629	233	231.499	1.00648	1.50577	18.0688
CCTTCTTCTTCA	35	21.3709	35	23.2529	1.50519	14.3122	17.2661
CTTCTTCTTTTT	62	46.987	63	51.1944	1.2306	13.0727	17.1904
ATCTCTCTCTCC	32	18.7936	32	20.4458	1.56511	14.3346	17.031
AGAGAAGAAGAA	97	81.7819	98	89.2702	1.09779	9.14342	16.5536

Table A.14: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATCTCTCTCTCTT	61	32.7842	61	35.9775	1.69551	32.2069	37.8765
CTTCTTCTTCCTC	156	129.249	157	142.574	1.10118	15.1326	29.3462
TCTTCTTCTTCCT	192	166.692	193	184.247	1.04751	8.95775	27.1393
TTCTCTCTCTCTT	70	48.4395	70	53.2022	1.31574	19.2077	25.7726
TCTCTCTCTCTTT	225	201.778	227	223.449	1.01589	3.5786	24.5102
CTCTCTTTCTCTC	135	112.702	136	124.211	1.09492	12.3321	24.3713
AAGAGAGAGAGAA	33	18.468	33	20.2512	1.62953	16.1136	19.1555
TTCTTCTTCTTCC	198	180.604	198	199.774	0.99112	-1.76618	18.2081
TCTCTCTTTCTCT	184	167.003	187	184.594	1.01303	2.42155	17.8343
TCTCTTTCTCTCT	175	158.077	180	174.644	1.03067	5.43723	17.7984
CTTCTTCTTCTCC	99	83.2635	99	91.6211	1.08054	7.66842	17.1379
CGAAGAAGAAGAC	18	7.0562	18	7.73281	2.32774	15.2082	16.8564
AAAAAAAAAACAG	51	36.7129	51	40.2973	1.26559	12.0126	16.7635
TTTCTCTCTCTCT	173	157.324	174	173.805	1.00112	0.195181	16.4327
CTCTCTCTCTTTC	165	149.773	167	165.397	1.00969	1.61107	15.9755
TTCTTCTTCTCCT	62	48.2157	62	52.9558	1.17079	9.77602	15.5899
AAAAAAAAAAACT	71	57.1005	71	62.7439	1.13158	8.77691	15.4685
CTCTCTCTTTCTC	147	132.412	148	146.087	1.01309	1.92517	15.3642
TCTTCTTCTTCAT	90	75.9711	90	83.564	1.07702	6.6777	15.2511
AAAAAAACAAAAA	60	46.5347	62	51.1049	1.21319	11.9818	15.2487
AAAGAAGAAGAAA	27	15.6991	27	17.2124	1.56863	12.1555	14.6403
TCTTCTTCTTCTC	206	191.93	208	212.431	0.97914	-4.38475	14.5739
ATCTTCTTCTTCA	31	19.4445	31	21.3231	1.45382	11.6	14.4592
CTTTCTTCTTCTT	91	77.8074	91	85.5923	1.06318	5.57505	14.2527
CCAAAAAAAAAAA	93	79.8196	94	87.8153	1.07043	6.39755	14.2132

Table A.15: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTTCTTCTTCCT	150	126.825	150	141.011	1.06375	9.26953	25.1746
CAAAAAAAAAAAAA	125	103.045	126	114.424	1.10117	12.143	24.1439
TCTCTCTCTCTCTT	199	179.559	200	200.216	0.998923	-0.215432	20.4576
TCTTCTTCTTCCTC	139	122.239	140	135.879	1.03033	4.18295	17.8609
CTTCTTCTTCTTCC	162	145.534	162	161.978	1.00014	0.0219795	17.3637
TTTCTTCTTCTTCT	140	124.521	141	138.432	1.01855	2.59143	16.4039
TCTCTTCTTCTTCT	109	94.0166	111	104.348	1.06375	6.85984	16.1185
TTTCTCTCTCTCTC	121	106.354	121	118.12	1.02438	2.91472	15.6107
TCTTTCTCTCTCTC	84	70.1935	84	77.8065	1.0796	6.43377	15.0832
CATCTCTCTCTCTC	58	45.1201	58	49.9459	1.16126	8.6711	14.5647
CTCTCTCTCTCTTC	69	56.7777	69	62.8901	1.09715	6.39758	13.4524
CTTCTTCTTCTCTC	55	43.4181	55	48.0575	1.14446	7.42146	13.0052
CTCTCTCTTTCTCT	117	105.151	118	116.776	1.01048	1.23002	12.4925
CTTCTTCTTCTTCA	133	121.437	134	134.982	0.992727	-0.978107	12.0967
CTCTCTTTCTCTCT	107	95.8873	108	106.435	1.01471	1.57667	11.7332
CCAAAAAAAAAAAA	64	53.2968	65	59.0233	1.10126	6.26962	11.7125
TTCTTCTTCTCTCT	59	48.4575	59	53.65	1.09972	5.6083	11.6142
GAGAGAGAGAGAGG	38	28.1844	38	31.1703	1.21911	7.52856	11.3551
CTTCTCTCTCTCTT	33	23.4728	33	25.953	1.27153	7.92727	11.2419
CAAAAAAAAAAAGG	12	4.70379	12	5.19553	2.30968	10.0453	11.2385
ATCTCTCTCTCTCT	137	126.25	137	140.368	0.976007	-3.32715	11.1952
CTCTCTCTCTCTTT	134	123.482	134	137.27	0.976181	-3.23038	10.9539
ATCTTCTTCTTCTC	24	15.2243	24	16.8255	1.42641	8.52383	10.9238
ACTCTCTCTCTCTC	76	65.8541	76	72.9793	1.04139	3.08237	10.8902
CTCTCTCTCTTTCT	112	101.64	113	112.855	1.00128	0.144624	10.8709

Table A.16: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTCTCTCTCTCTC	158	142.185	158	159.511	0.990529	-1.5036	16.6637
CTCTCTCTCTCTCTT	150	135.341	151	151.776	0.994885	-0.774365	15.4255
TTCTTCTTCTTCCTC	111	97.228	111	108.808	1.02014	2.21364	14.7043
TCTTCTTCTTCTTCC	133	120.729	133	135.282	0.983131	-2.26273	12.8742
CTTCTTCTTCTTCCT	121	109.592	121	122.727	0.985926	-1.71506	11.9825
CCAAAAAAAAAAAAA	56	45.3031	56	50.5556	1.10769	5.72762	11.8706
CTTCTTCTTCTTCGT	47	36.5671	47	40.7872	1.15232	6.66359	11.797
TCTCTCTCTCTCTTC	64	53.5097	64	59.7403	1.0713	4.40812	11.4573
AAGAAGAAGAAGAAA	73	62.4443	73	69.7492	1.04661	3.32544	11.4015
TCTCTCTCTCTTTCT	91	80.4896	91	89.994	1.01118	1.01159	11.1686
CCTCTCTCTCTCTCT	73	62.9517	73	70.3179	1.03814	2.73257	10.8106
ACCAAAAAAAAAAAA	43	34.1014	43	38.0319	1.13063	5.2793	9.97
AAAAAAAAAAAAAAG	64	55.0318	64	61.4447	1.04159	2.60767	9.66219
CTCTTCTTCTTCTTC	118	108.754	119	121.784	0.977141	-2.75185	9.62807
TCTCTCTCTCTCTCG	64	55.2373	64	61.6749	1.0377	2.36838	9.42362
GTTCTTCTTCTTCTC	9	3.17286	9	3.5326	2.5477	8.41672	9.38333
AAGAGAGAGAGAAAA	10	3.91911	10	4.36364	2.29167	8.29279	9.36721
TTTTCTTCTTCTTCT	50	41.4676	50	46.2656	1.08072	3.88119	9.35557
TCTTCTTCTTCCTCT	63	54.3105	63	60.6369	1.03897	2.40852	9.35033
CTCTCTCTCTCTCCA	20	12.5679	20	14	1.42857	7.1335	9.29176
TTCTTCTTCTTCATC	38	29.7859	38	33.2111	1.1442	5.11866	9.255
TTTCTTCTTCTTCTT	112	103.13	112	115.45	0.970113	-3.39838	9.24087
TTTCTCTCTCTCTCT	94	85.3057	94	95.4038	0.985285	-1.39347	9.12305
CAAAAAAAAAACAA	10	4.04155	10	4.5	2.22222	7.98508	9.05956
TCTCTCTCTCTCTCC	66	57.564	66	64.2809	1.02674	1.7419	9.02599

Table A.17: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAAAA	70	55.2254	70	62.1657	1.12602	8.30838	16.595
CTCTCTCTCTCTCTCA	63	50.7804	64	57.1482	1.1199	7.24705	13.5843
TTCTCTCTCTCTCTCT	131	118.411	131	133.756	0.979395	-2.7274	13.2359
TCTTCTTCTTCTTCCT	100	88.0896	100	99.3395	1.00665	0.662685	12.6815
TCTCTCTCTCTCTCTT	128	116.046	129	131.068	0.984225	-2.05114	12.5497
ATCTCTCTCTCTCTCT	89	77.3664	89	87.1954	1.0207	1.82319	12.4675
TTTCTTCTTCTTCTTC	99	88.0191	99	99.2596	0.997385	-0.259226	11.6391
TCTTCTTCTTCTTCTC	95	84.6685	95	95.4635	0.995145	-0.462388	10.9376
TTTCTCTCTCTCTCTC	74	64.3282	74	72.4488	1.02141	1.56771	10.3649
AAAAAAAAAAAAAAAC	46	36.8541	46	41.4438	1.10994	4.79791	10.197
AAGAGAGAGAGAGAGA	64	54.6184	65	61.4804	1.05725	3.61843	10.1447
TCTCTTCTTCTTCTTC	76	66.6044	77	75.0217	1.02637	2.00412	10.0291
CTCTCTCTCTCTCTTC	52	42.9544	52	48.32	1.07616	3.8167	9.93743
TCTCTCTCTCTCTCTA	67	57.8236	67	65.0998	1.02919	1.9277	9.86879
CTTCTCTCTCTCTCTC	59	49.9964	59	56.2634	1.04864	2.80208	9.7696
CTCTTCTTCTTCTTCC	20	12.5943	20	14.1439	1.41404	6.92901	9.24977
CTTCTTCTTCTTCCTC	88	79.4377	88	89.54	0.982801	-1.52668	9.00802
TCTTCTTCTTCTTCAT	45	36.8907	45	41.4851	1.08473	3.65979	8.94158
CCACACACACACACAT	3	0.159591	3	0.179104	16.75	8.45519	8.80126
TCTCTCTCTCTCTTTC	64	55.9246	64	62.9552	1.0166	1.0534	8.63224
TTCTCTCTCTCTTTCT	26	18.8802	26	21.2105	1.22581	5.29357	8.31954
CCTCTCTCTCTCTCTC	58	50.6784	58	57.033	1.01696	0.975151	7.82675
TATCTCTCTCTCTCTC	35	28.0932	35	31.5766	1.10841	3.60256	7.6937
TCTCTCTCTCTTTCTC	55	47.9512	55	53.9558	1.01935	1.05429	7.54324
CCAAAAAAAAAAAAA	41	34.3892	41	38.6667	1.06034	2.40236	7.20902

Table A.18: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTCTCTCTCTCTCTCTT	13	4.23278	13	4.79045	2.71373	12.9782	14.5872
TTCTCTCTCTCTCTCTA	16	6.85598	16	7.76039	2.06175	11.5769	13.5595
ATCTCTCTCTCTCTCTC	73	61.2712	73	69.5632	1.04941	3.52031	12.786
TTTCTCTCTCTCTCTCT	64	54.0625	64	61.3544	1.04312	2.7018	10.7995
CAAAAAAAAAAAAAAAA	53	43.2659	53	49.0722	1.08004	4.081	10.7552
ATTCTTCTTCTTCTTCG	6	1.10854	6	1.25438	4.78326	9.39073	10.1323
AGAAGAAGAAGAAGAAA	50	40.932	50	46.4191	1.07714	3.71558	10.0055
TCTCTCTCTCTCTCTTT	68	58.701	68	66.6358	1.02047	1.37811	9.99944
AAAAAAAAAAAAAAAAG	37	28.4552	37	32.2474	1.14738	5.08675	9.71572
CTCTTCTTCTTCTTCTT	68	59.1138	68	67.1058	1.01332	0.900091	9.52298
CTCTCTCTCTCTCTCTG	42	33.5779	42	38.0637	1.10341	4.1332	9.39964
TTCTTCTTCTTCTTCTC	74	65.7435	74	74.6594	0.991168	-0.656447	8.75449
CTTCTTCTTCTTCTTCA	61	53.0817	61	60.238	1.01265	0.766759	8.48159
TCTTCTTCTTCTTCCTC	72	64.0482	72	72.7273	0.99	-0.723624	8.42619
CTTTCTCTCTCTCTCTC	40	32.6413	40	37	1.08108	3.11846	8.13204
AAGAGAGAGAGAGAGAG	52	45.2876	53	51.371	1.03171	1.65459	7.18697
TCTTCTTCTTCTTCTTT	49	42.5689	49	48.2799	1.01492	0.725494	6.89409
GAAAAAAAAAAAAAAAT	5	1.30965	5	1.48196	3.37391	6.08037	6.69838
CTTTTTTTTTTTTTTTG	5	1.32559	5	1.5	3.33333	6.01986	6.63789
TTTCTTCTTCTTCTTCT	72	65.7138	72	74.6254	0.964818	-2.5787	6.57777
CTTCTTCTTCTTCTTCC	77	70.7064	77	80.3174	0.958697	-3.24791	6.56575
TTCTTCTTCTTCTTCAT	32	26.2324	32	29.7248	1.07654	2.36017	6.35966
CTTTCTTCTTCTTCTTC	33	27.3011	33	30.9375	1.06667	2.12977	6.2562
CTTCTCTCTCTCTCTCT	49	43.13	49	48.9177	1.00168	0.0823476	6.25247
ATCTCTCTCTTTCTCTG	6	2.12086	6	2.4	2.5	5.49774	6.23963

Table A.19: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTCTCTCTCTCTCTCG	12	3.82062	12	4.35973	2.75246	12.15	13.7339
ATCTCTCTCTCTCTCTCA	12	3.83549	12	4.3767	2.74179	12.1033	13.6873
TTTCTTCTTCTTCTTCTC	15	7.22643	15	8.24768	1.81869	8.97178	10.9546
TTCTCTCTCTCTCTCTCC	10	3.44795	10	3.93439	2.54169	9.32829	10.648
TCTTCTTCTTCTTCTTCA	54	45.0771	54	51.5565	1.04739	2.50053	9.75286
CTTCTTCTTCTTCTTCCT	57	48.3795	57	55.3437	1.02993	1.68079	9.34656
CTCTTCTTCTTCTTCTTC	60	52.8832	60	60.511	0.991555	-0.508849	7.5755
GTTTCTCTCTCTCTCTCA	3	0.248742	3	0.283784	10.5714	7.07446	7.46985
CTCTCTCTCTCTCTCTTC	33	26.5622	33	30.3488	1.08736	2.76373	7.16155
TTCTTCTTCTTCTTCCTC	50	43.4405	50	49.68	1.00644	0.321028	7.03158
TTTCTTCTTCTTCTTCTG	5	1.26993	5	1.44892	3.45085	6.19311	6.85238
CCTTCTTCTTCTTCTTCT	35	28.8131	35	32.9247	1.06303	2.13939	6.80814
AAAGAGAGAGAGAGAGAG	34	27.8304	35	31.8	1.10063	3.35586	6.8079
CTTTCTTCTTCTTCTTCT	27	21.0121	27	24	1.125	3.18014	6.76998
ATCTCTCTCTCTCTCTCC	7	2.67778	7	3.05543	2.291	5.80293	6.72645
CTCTCTCTCTTTCTCTTT	13	7.80569	13	8.90909	1.45918	4.9124	6.63125
GCTTCTTCTTCTTCTTCC	7	2.82353	7	3.22176	2.17273	5.43188	6.35546
TTCTTCTTCTTCTTCTCT	33	27.2693	33	31.1579	1.05912	1.89552	6.29464
CTTTCTCTCTCTCTCTCT	36	30.272	36	34.5946	1.04063	1.43357	6.23866
TTCTCTTTCTCTCTCTCT	12	7.27899	12	8.30769	1.44444	4.4127	5.99897
TCTCTTCTTCTTCTTCTT	43	37.4501	43	42.8148	1.00433	0.185585	5.94225
CTTCTCTCTCTCTCTCTT	10	5.57193	10	6.35878	1.57263	4.52749	5.84843
AAGAAGAAGAAGAAGAAA	41	35.5639	41	40.6542	1.00851	0.347261	5.83185
TCTCTCTCTCTCTCTCTT	72	66.4454	73	76.0871	0.959427	-3.02362	5.78054
CTCTCTCTTCTTCTTCTT	14	9.38648	14	10.7143	1.30667	3.74471	5.59701

Table A.20: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CCTTCTTCTTCTTCTTCTT	35	25.8977	35	29.8361	1.17308	5.58706	10.5418
ACTTCTTCTTCTTCTTCTC	8	2.17686	8	2.50455	3.19418	9.29065	10.4125
CTCTTCTTCTTCTTCTTCT	49	39.9308	49	46.0396	1.0643	3.0536	10.029
GCTCTCTCTCTCTCTCTCG	4	0.349991	4	0.402635	9.93455	9.18407	9.74456
CCTCTCTCTCTCTCTCTCA	6	1.38461	6	1.59297	3.76654	7.95695	8.79803
ATCTTCTTCTTCTTCTTCC	8	3.37889	8	3.88779	2.05772	5.77281	6.89516
TTTCTCTCTCTCTCTCTCT	34	27.8376	34	32.0745	1.06003	1.98221	6.79909
CTTCTTCTTCTTCTTCCTC	42	35.8321	42	41.3043	1.01684	0.701478	6.67068
TCTCTTCTTCTTCTTCTTC	39	32.9199	39	37.9412	1.02791	1.07346	6.60989
TTTTCTTCTTCTTCTTCTT	20	14.476	20	16.6667	1.2	3.64643	6.46482
TCTCTCTCTCTCTCTCTTT	37	31.3083	37	36.0805	1.02549	0.93116	6.1803
AGAGAGAGAGAGAGAGAAA	21	15.7102	21	18.0889	1.16093	3.13372	6.09448
ATCTCTCTCTCTCTCTCTA	6	2.1781	6	2.50598	2.39428	5.23849	6.07984
TTCTCTCTCTCTCTTTCTT	5	1.49005	5	1.71429	2.91667	5.35221	6.05314
CTTCTTCTTCTTCTTCTTT	35	29.4416	35	33.9257	1.03167	1.09118	6.0529
AATCTCTCTCTCTCTCTCT	14	9.10478	14	10.4795	1.33595	4.05497	6.0236
GTCTCTCTCTCTCTCTTTG	4	0.920354	4	1.05882	3.77778	5.31654	5.87716
TCTTCTTCTTCTTCTTCAT	26	20.7526	26	23.9016	1.08779	2.18789	5.86099
TTTCTTCTTCTTCTTCTCT	10	5.8128	10	6.68919	1.49495	4.02092	5.42523
GAGAGAGAGAGAGAGAGAT	25	20.3213	25	23.4043	1.06818	1.64895	5.18017
CTTCTTCTTCTTCTTCTCC	22	17.427	22	20.0676	1.0963	2.02262	5.12651
CTCTCTCTCTCTCTCTCCT	7	3.38245	7	3.89189	1.79861	4.1091	5.09116
ACTCTCTCTCTCTCTCTCC	3	0.572706	3	0.658858	4.55333	4.54758	4.96798
CTCTCTTTCTCTCTCTCTC	12	7.94864	12	9.14815	1.31174	3.25626	4.94288
TCCTCTCTCTCTCTCTCTT	6	2.6961	6	3.10204	1.93421	3.9582	4.79973

Table A.21: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TCTTCTTCTTCTTCTTCCTC	35	27.9028	35	32.4211	1.07955	2.6789	7.93174
TTTCTTCTTCTTCTTCTTCG	6	1.62488	6	1.88518	3.18272	6.94643	7.83794
CTCTTCTTCTTCTTCTTCTT	43	36.0782	43	41.9398	1.02528	1.0735	7.54701
CCTTCTTCTTCTTCTTCTTC	33	26.9057	33	31.2607	1.05564	1.78683	6.73764
TTTCTTCTTCTTCTTCTTCC	8	3.63623	8	4.21921	1.89609	5.11836	6.30796
AAGAAGAAGAAGAAGAAGAG	17	11.7431	17	13.6321	1.24706	3.75339	6.28914
TCTCTTCTTCTTCTTCTTCT	33	27.4121	33	31.85	1.03611	1.17052	6.12227
TGAGAGAGAGAGAGAGAGAT	4	0.914201	4	1.06061	3.77143	5.30982	5.904
TTCTTCTTCTTCTTCTTCTC	35	29.6031	35	34.4	1.01744	0.605202	5.86143
ACTCTCTCTCTCTCTCTCTT	5	1.56456	5	1.81518	2.75455	5.06626	5.80917
CTTTCTCTCTCTCTCTCTCT	22	16.9173	22	19.6444	1.11991	2.49145	5.77948
TTCTTCTTCTTCTTCTTCCT	34	28.7051	34	33.3548	1.01934	0.651361	5.75567
CTTCTTCTTCTTCTTCCTCT	21	16.6382	21	19.32	1.08696	1.75101	4.88925
ATCTTCTTCTTCTTCTTCTC	5	1.91289	5	2.21935	2.25291	4.06111	4.80412
TTCTCTCTCTCTCTCTCTCT	44	39.5631	44	46	0.956522	-1.95588	4.6769
CAAAAAAAAAAAAAAAAAAG	6	2.76884	6	3.2126	1.86765	3.74808	4.63998
CTCTCTCTCTCTCTCTCTTC	21	16.8697	21	19.589	1.07203	1.4606	4.59911
TTCTCTCTCTCTCTCTCTCA	5	2.01112	5	2.33333	2.14286	3.8107	4.55374
GTCTTCTTCTTCTTCTTCTT	18	14.0071	18	16.2624	1.10685	1.82733	4.51455
CAGAGAGAGAGAGAGAGAGA	19	15.0404	19	17.4631	1.08801	1.6026	4.44023
TCTTCTTCTTCTTCTTCTCT	18	14.1412	18	16.4182	1.09635	1.65568	4.34305
TCAAAAAAAAAAAAAAAAAA	9	5.58961	9	6.48649	1.3875	2.94753	4.28683
TTCTTCTTCTTCTTCTTCAT	20	16.1717	20	18.7778	1.06509	1.26116	4.24943
TAAAAAAAAAAAAAAAAAAA	8	4.77706	8	5.54331	1.44318	2.9348	4.12492
AAGAGAGAGAGAGAGAGAGA	30	26.3039	30	30.5605	0.98166	-0.555301	3.9444

A.2 The *Arabidopsis thaliana* 3' Untranslated Regions

A.2.1 Timing Information

Table A.22: Full Size and Timing Information, 5' UTR

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	0.555072	1.096124	7.987123	1.651196
2	0	0.000001	0.988411	1.223718	14.580604	2.212129
3	1	0.000003	1.539554	1.151038	10.416979	2.690592
4	2	0.000012	2.339968	1.193031	3.913065	3.532999
5	3	0.000046	3.152420	1.176259	1.942146	4.328679
6	4	0.000186	4.092285	3.004922	1.404120	7.097207
7	5	0.000743	5.149271	6.622906	1.277710	11.772177
8	6	0.002961	6.286397	21.548922	1.499570	27.835319
9	7	0.011393	8.400295	66.231518	2.810946	74.631813
10	8	0.038819	10.780911	201.141277	6.734395	211.922188
11	9	0.108122	13.259231	464.514664	14.772724	477.773895
12	10	0.239873	16.058718	483.208583	25.640569	499.267301
13	11	0.433401	18.752481	689.538892	36.289080	708.291373
14	12	0.668341	21.622854	1017.216637	44.523756	1038.839491
15	13	0.923532	24.354624	1090.296268	50.821547	1114.650892
16	14	1.186549	27.995046	823.104031	56.562577	851.099077
17	15	1.451913	30.100335	1115.968849	61.897430	1146.069184
18	16	1.717478	33.896624	1108.682488	67.199997	1142.579112
19	17	1.982418	36.459362	805.643609	72.307066	842.102971
20	18	2.246407	40.244915	1104.470917	77.921645	1144.715832

A.2.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.23: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
C	20006	20003.2	691932	691932	1	0	2.81981
G	19999	20010.5	805619	805619	1	0	-11.4497
T	20028	20034.4	1710075	1.71008e+06	1	0	-6.42759
A	20027	20028.4	1348467	1.34847e+06	1	0	-1.35685

Table A.24: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CC	18549	19268.6	105789	105083	1.00671	707.954	-706.017
AC	19792	19814.2	193430	204791	0.944524	-11040	-22.1736
CG	17452	19467.6	73089	122349	0.597381	-37655.4	-1907.47
CA	19838	19814.2	233963	204791	1.14245	31157.5	23.8283
GC	18806	19467.6	107250	122349	0.876591	-14126.4	-650.248
AG	19784	19862.1	243131	238439	1.01968	4737.83	-77.9436
AT	19971	19974.8	438889	506131	0.867145	-62563.1	-3.82865
GA	19845	19862.1	263671	238439	1.10582	26522.3	-17.09
GG	18729	19612.5	134681	142451	0.945452	-7554.51	-863.268
GT	19877	19913.7	297377	302379	0.983457	-4960.78	-36.7069
TG	19926	19913.7	350616	302379	1.15952	51894.4	12.263
TC	19871	19883.3	283166	259708	1.09032	24486.5	-12.3221
CT	19866	19883.3	273538	259708	1.05325	14191.5	-17.3184
AA	19948	19952.5	468205	399106	1.17314	74762.9	-4.54402
TA	19942	19974.8	375064	506131	0.741042	-112406	-32.802
TT	19971	19990.7	694184	641856	1.08153	54405.3	-19.7207

Table A.25: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTA	18553	18111	71712	65222.5	1.0995	6802.12	447.391
ATA	19335	19103.6	115180	96259.8	1.19655	20668.5	232.747
AAC	18369	18207.5	70535	67161.4	1.05023	3456.98	162.225
ATG	19101	18973.7	96406	89985.2	1.07135	6644.57	127.71
TAT	19571	19449	141318	122073	1.15765	20688	122.387
TAC	17458	17381.6	55980	53800.8	1.0405	2222.73	76.5282
TAA	19592	19517.6	116755	130227	0.89655	-12749.8	74.5723
TGT	19577	19511.4	146880	129422	1.13489	18585.4	65.7155
AAT	19702	19649.5	133188	152388	0.874007	-17936.1	52.5438
TTG	19632	19597.6	148503	142328	1.04338	6306.77	34.3853
CAT	18608	18579.2	78076	76148.5	1.02531	1951.66	28.7952
GTT	19450	19436.2	126144	120717	1.04496	5547.67	13.8322
CTT	19326	19330.4	117672	111039	1.05973	6826.86	-4.36551
ATT	19727	19742.8	152304	178162	0.854864	-23883.2	-15.7725
TTA	19633	19648.9	133467	152253	0.876615	-17575.9	-15.8936
CAA	18729	18744.9	84373	81234.9	1.03863	3197.9	-15.9173
ATC	18361	18448.9	73738	72674.3	1.01464	1071.48	-87.6453
ACA	18013	18120.3	72684	65404.5	1.1113	7670.36	-106.985
TTT	19772	19890.8	295826	281795	1.04979	14374.1	-118.47
TTC	19251	19376.4	113135	114948	0.98423	-1798.4	-124.956
TCT	19193	19341.4	124789	111943	1.11476	13556.8	-147.856
ACC	13987	14169.8	32137	29573.4	1.08669	2671.66	-181.592
CCA	15128	15320.1	39835	35770.4	1.11363	4287.21	-190.92
GAT	18670	18872.6	84462	85817.7	0.984203	-1344.91	-201.485
GAA	18787	19008.5	92376	91549.9	1.00902	829.77	-220.162

Table A.26: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TGTA	16547	15225.5	39771	35419.9	1.12284	4608.05	1377.31
GTAA	13511	12312.5	25417	22323.5	1.13858	3298.6	1255.06
TAAT	16010	14844.6	39001	33212.7	1.17428	6265.67	1209.98
ACTA	9930	8888.74	15625	13105.1	1.19229	2748.03	1100
TAGT	12059	11026.6	21518	18361.1	1.17193	3413.91	1079.31
ATTA	15062	14069.3	34091	29282.7	1.1642	5183.12	1026.92
AAAC	14086	13286.2	29392	25906	1.13456	3710.63	823.375
ATCA	13088	12320.5	26300	22350.9	1.17669	4279.11	790.882
TTAC	12235	11561.8	21371	19920.6	1.07281	1502.01	692.404
CTGC	5222	4607.85	7379	5574.08	1.3238	2069.88	653.375
GATT	14671	14075.2	32614	29310.1	1.11272	3483.43	608.226
ATAA	15882	15296.4	37613	35854.8	1.04904	1800.63	596.72
CTAC	6688	6168.4	9436	7978.25	1.18272	1583.48	540.898
TGAT	15037	14543.4	35730	31604.5	1.13054	4383.76	501.912
CATT	14062	13575.4	28389	27094.1	1.04779	1325.4	495.205
AATC	12795	12328.2	23862	22377	1.06636	1533.23	475.494
AATG	14513	14063.6	29916	29256	1.02256	667.422	456.535
TACT	11797	11390.4	20725	19408.2	1.06785	1360.48	413.794
AATA	15551	15147.9	35846	34953.2	1.02554	904.062	408.41
TTAG	12332	11989.6	23023	21256.4	1.08311	1838.04	347.208
ATGA	13924	13583.5	29619	27128.3	1.09181	2601.71	344.718
TAAA	16616	16279.9	41066	42881.7	0.957658	-1776.7	339.545
TTTG	18308	17978.8	68729	63284.4	1.08603	5672.32	332.17
CGTC	3356	3050.61	4260	3466.43	1.22893	878.173	320.193
TCAT	14243	13929.9	30439	28642.7	1.06271	1851.44	316.63

Table A.27: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AATAA	9290	8186.94	14627	11705.8	1.24955	3258.65	1174.24
TGAAT	7659	6739.87	10341	8991.01	1.15015	1446.61	979.136
ATGTA	8495	7573.11	11895	10506.5	1.13216	1476.46	975.852
TAAAT	8625	7817.32	12104	10974.7	1.1029	1185.52	848.035
ACTTG	5236	4457.8	6426	5385.52	1.1932	1135.08	842.487
ATAAA	9692	8913.83	14343	13229.5	1.08416	1159.05	811.192
TGTAT	9779	9066.8	14434	13565.9	1.06399	895.292	739.463
GAATC	4443	3765.73	5466	4423.11	1.23578	1157.18	734.821
GATTC	4931	4264.5	6194	5111.25	1.21184	1190.1	716.06
GTAAC	3686	3041.87	4216	3471.8	1.21435	818.801	707.967
TGTAA	9984	9302.7	14487	14096.1	1.02773	396.271	705.658
ATTTA	8740	8124.59	12479	11580.5	1.07758	932.457	638.147
ATTCA	6087	5495.91	7912	6936.75	1.14059	1040.81	621.796
GTTTC	7710	7122.72	10631	9672.21	1.09913	1004.82	610.848
ATCTA	4751	4186.09	5815	5001.22	1.16272	876.661	601.414
TCAAT	6656	6089.03	8529	7887.48	1.08133	666.924	592.586
ATTTG	10134	9567.66	15374	14709	1.04521	679.821	582.779
TATGT	9061	8506.92	13430	12361.9	1.0864	1112.93	571.745
CAAGT	4408	3873.28	5381	4569.16	1.17768	880.036	570.045
ATACA	5200	4683.16	6675	5710.86	1.16883	1041.3	544.361
CTTAA	5324	4812.99	6453	5901.07	1.09353	576.977	537.222
AGTAG	3031	2542.35	3721	2845.9	1.30749	997.647	532.856
ATTGA	6482	5981.82	8478	7711.9	1.09934	802.952	520.537
TTAAG	5709	5216.32	7182	6505.42	1.104	710.606	515.245
ATAGA	4828	4347.43	5937	5228.39	1.13553	754.599	506.198

Table A.28: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTG	4755	4160.29	5682	4989.47	1.1388	738.514	635.328
GTTTTG	5147	4589.77	6377	5602.8	1.13818	825.38	589.765
ATTTTG	5320	4781.61	6432	5883.9	1.09315	572.875	567.617
ATTTTA	4395	3863.93	5257	4578.6	1.14817	726.344	565.998
TAAAAT	3757	3238.25	4337	3742.48	1.15886	639.419	558.247
TTTTGT	8993	8507.04	12879	12428	1.03629	459.12	499.579
AATATT	3613	3172.37	4110	3656.82	1.12393	480.165	469.91
TTATAA	3500	3067.99	3912	3522.01	1.11073	410.832	461.095
TGAAAT	3113	2685.62	3462	3037.35	1.13981	453.037	459.715
GATTTG	3242	2823.52	3694	3210.5	1.1506	518.212	448.064
ATTTGA	3632	3221.13	4212	3720.19	1.1322	522.973	436.021
TGTAAT	4545	4138.39	5270	4958.76	1.06276	320.805	425.966
AATAAA	4964	4572.51	6151	5577.73	1.10278	601.772	407.796
ATTTCA	2808	2429.34	3108	2720.42	1.14247	413.97	406.743
ATTTTC	3859	3473.1	4465	4051.47	1.10207	433.951	406.591
GTTTTA	4056	3703.23	4677	4359.87	1.07274	328.398	369.065
TGTTTT	8211	7853.56	11460	11103.1	1.03214	362.548	365.458
ACAAAA	3682	3357.01	4278	3898.03	1.09748	397.918	340.234
ATGTTA	2758	2438.53	3036	2731.67	1.11141	320.685	339.537
GTAATG	2046	1738.28	2209	1895.88	1.16516	337.658	333.474
TTAATT	3988	3683.23	4862	4332.84	1.12213	560.229	317.045
TATGTA	3826	3522.39	4497	4117.05	1.09229	396.964	316.337
CATTTT	4294	4004.83	4991	4772.72	1.04574	223.199	299.364
TACATA	2136	1858.04	2413	2035.71	1.18533	410.269	297.782
TTTGAT	5287	4999.73	6551	6209.03	1.05508	351.219	295.366

Table A.29: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTG	2129	1614.09	2322	1760.4	1.31902	642.937	589.477
ATTTTTA	1820	1342.1	1935	1448.82	1.33557	559.908	554.378
TTTTGTT	5034	4625.79	6145	5683.31	1.08124	479.956	425.715
GTTCTTG	943	620.803	988	652.385	1.51444	410.068	394.223
TTTGTTT	5098	4727.58	6495	5833.14	1.11347	698.07	384.566
TTTTTGT	4168	3812.13	4893	4529.82	1.08018	377.367	371.99
ATTTTTG	1911	1606.05	2030	1751.1	1.15927	300.018	332.22
GTTTTTA	1644	1348.88	1737	1456.51	1.19257	305.91	325.278
TTTATTT	3317	3014.84	4036	3470.41	1.16297	609.361	316.82
TAAAAAT	1262	999.845	1339	1065.61	1.25655	305.792	293.86
TTTTATT	3130	2850.56	3593	3260.24	1.10207	349.197	292.71
AATAAAA	2383	2109.18	2647	2344.1	1.12922	321.682	290.873
TGTTTTT	3937	3659.2	4588	4321.4	1.06169	274.663	288.083
GATTTTG	1483	1227.2	1575	1319.08	1.19401	279.274	280.781
CAAAAAC	796	562.007	824	589.313	1.39824	276.215	277.076
ATTTTTC	1552	1301.57	1648	1402.93	1.17468	265.327	273.113
CTTTTTA	1512	1264.41	1581	1360.98	1.16166	236.913	270.384
ATTTTGA	1557	1309.85	1644	1412.29	1.16407	249.755	269.13
ATTTGTA	1710	1461.71	1811	1585.06	1.14255	241.333	268.281
ATTTTCA	1305	1080.94	1349	1155.53	1.16743	208.831	245.83
TTTTCTT	4225	3989.04	5157	4774.16	1.08019	397.798	242.802
TTTCATT	1893	1666.36	2022	1821	1.11038	211.705	241.397
CTTCTTC	1405	1193.25	1833	1280.96	1.43096	656.844	229.52
TGTTTGT	2497	2281.88	2915	2552.89	1.14184	386.653	224.958
AAAGAAA	2082	1869.77	2404	2059.13	1.16749	372.264	223.838

Table A.30: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTGTT	2292	2092.08	2520	2334.61	1.07941	192.56	209.182
TTTTTTGG	1010	832.111	1060	885.402	1.1972	190.782	195.677
TTTTTCTT	2192	2005.96	2428	2231.13	1.08824	205.311	194.416
ATTTTGTA	739	589.001	759	621.081	1.22206	152.21	167.657
TAATTTTT	794	647.449	817	684.196	1.1941	144.932	162.011
ATGTTTTA	595	473.658	607	497.326	1.22053	120.965	135.705
ATTTTTTA	491	391.156	507	409.449	1.23825	108.346	111.623
GTTTTTGA	495	395.411	507	413.968	1.22473	102.78	111.193
TTTGTTTT	2540	2433.27	2884	2751.3	1.04823	135.851	109.039
AAATTTTG	593	495.756	608	520.954	1.16709	93.9443	106.213
ATTTTTCA	446	356.258	450	372.439	1.20825	85.1289	100.2
ATAAAAAT	572	480.167	589	504.281	1.168	91.4671	100.103
TGTTTTGT	1242	1146.41	1341	1234.25	1.08649	111.241	99.47
AAAAATTG	401	314.407	405	328.178	1.23408	85.1835	97.5528
TTTTTAAT	907	815.38	939	867.058	1.08297	74.8468	96.5847
ATTTTCTG	331	247.645	338	257.857	1.31081	91.4771	96.0299
CTCTGTTT	771	682.009	822	721.643	1.13907	107.032	94.5599
TATAATAT	510	424.298	524	444.686	1.17836	86.0003	93.8267
TTTTTGGT	1000	912.352	1040	973.691	1.0681	68.5171	91.7299
TTCTTTTT	1902	1813.32	2093	2002.09	1.04541	92.943	90.8128
TTTTTTCT	1734	1646.71	1855	1806.65	1.02676	48.99	89.5684
TAAGAAAT	302	227.421	306	236.621	1.2932	78.6797	85.6548
AATATATT	463	385.27	475	403.201	1.17807	77.8433	85.0908
TGTTTTTT	1727	1644.86	1851	1804.5	1.02577	47.0917	84.1556
ATTTTTAT	757	677.627	787	716.89	1.0978	73.4314	83.8501

Table A.31: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTCTCTT	258	177.354	262	185.033	1.41596	91.1265	96.701
TTTTTTCTT	1102	1009.5	1155	1086.33	1.06321	70.792	96.6167
TCTTTTTTT	1016	925.083	1068	992.35	1.07623	78.4634	95.2454
TTTTTCTTT	1159	1080.41	1243	1165.74	1.06627	79.7641	81.3825
ATTTTTTTA	254	184.498	257	192.536	1.33481	74.2195	81.2033
TTTTTTTAA	539	465.512	549	490.878	1.1184	61.4346	79.0054
TTTTTGTTT	1235	1162.1	1322	1257.74	1.05109	65.8696	75.1368
CTTTTTTTC	315	248.326	323	259.758	1.24346	70.382	74.9163
TTTTTTTCT	1014	941.965	1059	1011.1	1.04738	49.0182	74.7213
TTGTTTCTT	534	465.741	544	491.124	1.10766	55.6254	73.0328
TTTTTTGTT	1001	931.665	1041	999.657	1.04136	42.1867	71.8535
ATTTTCTTG	188	130.613	189	136.033	1.38937	62.1531	68.4705
ATTTTTTTC	282	224.372	283	234.493	1.20686	53.21	64.4662
AAAAGAAAA	518	457.476	559	482.26	1.15913	82.5454	64.3623
AGTTTTTTT	508	448.229	522	472.35	1.10511	52.1722	63.5898
ATTCTTTTG	237	184.507	239	192.546	1.24126	51.6545	59.3383
TTCTTTTTT	923	866.526	970	927.5	1.04582	43.4592	58.2758
TTTGTTTTT	1146	1089.74	1219	1176.22	1.03637	43.5468	57.6876
ATTTTTTTG	308	256.14	311	268.009	1.16041	46.2677	56.7875
TTTTTTTGT	1044	989.23	1099	1063.71	1.03317	35.8643	56.2592
TCTTTTGTT	544	491.035	551	518.283	1.06312	33.7281	55.7235
TTTTGTTTT	1308	1253.95	1453	1361.85	1.06693	94.1325	55.1995
TTCTTTTTC	429	377.244	444	396.5	1.1198	50.2379	55.1539
TTTGTGTTT	511	459.614	527	484.552	1.0876	44.2551	54.1575
GTTTGTTTG	319	269.654	364	282.291	1.28945	92.5348	53.6084

Table A.32: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTCTTTTTT	270	207.593	273	217.821	1.25332	61.6425	70.9676
GATTTTTTTT	232	186.826	234	195.88	1.19461	41.6091	50.2416
GTTTTTTTTT	744	695.852	777	743.532	1.04501	34.2107	49.7765
TTTTTTTCTT	666	618.872	689	659.377	1.04493	30.2783	48.8788
AGTTTTTTTT	339	295.55	346	311.126	1.11209	36.7595	46.4981
TTCTTCTTTT	342	298.895	347	314.687	1.10268	33.9183	46.0733
TTTTTGTTTG	304	261.566	310	275.004	1.12726	37.1343	45.7039
TTTTTTCTTT	600	556.27	618	591.295	1.04516	27.2988	45.4057
CTTTTTTTTT	696	652.888	722	696.505	1.0366	25.9563	44.5046
TTTTTCTTTC	274	233.259	277	244.985	1.13068	34.0208	44.1078
ATTTTTTTTA	122	87.7456	123	91.661	1.3419	36.1727	40.2087
TTTTTTGTTC	170	134.614	170	140.865	1.20683	31.9588	39.6753
TTTTTGTGTG	139	104.552	139	109.285	1.27191	33.4317	39.5864
TTTTTTTTGT	627	588.673	654	626.495	1.0439	28.0999	39.5488
TTTTGTTGTT	290	254.923	293	267.953	1.09347	26.1825	37.3868
TTTTCTTTTC	258	223.206	286	234.339	1.22045	56.9778	37.3754
TTTCTTCTTT	276	241.181	280	253.38	1.10506	27.9718	37.2193
CTTTTTTTCT	204	170.29	207	178.433	1.1601	30.7401	36.8459
TTCTTTTTTC	168	134.919	170	141.186	1.20409	31.5725	36.8405
TTTCTTTTTC	249	215.455	257	226.137	1.13648	32.8797	36.0306
TTTCTTTTTG	221	188.469	225	197.615	1.13858	29.2005	35.1897
TTTTTTTGGT	303	269.841	306	283.792	1.07826	23.0556	35.1172
ATTTTTTTTC	135	104.091	135	108.802	1.24079	29.1259	35.1011
GTTTTTGTTT	314	281.213	323	295.876	1.09167	28.3307	34.6278
TTGTTTTGTT	342	309.796	361	326.295	1.10636	36.4881	33.8227

Table A.33: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTT	502	467.204	517	497.28	1.03966	20.1059	36.0604
CTTCTTTTTTT	175	144.248	175	151.698	1.15361	25.0066	33.819
TTTTTTTTTAA	208	178.512	213	187.97	1.13316	26.6267	31.7999
GATTTTTTTTT	150	121.383	151	127.544	1.18391	25.4916	31.7522
TTCTTCTTTTT	189	159.802	192	168.152	1.14182	25.4639	31.7163
TTTTGTTGTTG	139	111.759	139	117.389	1.18409	23.4879	30.3197
TTATATATATT	59	36.2417	59	37.9609	1.55423	26.0178	28.7523
TTTCTTTTTCT	171	145.848	174	153.39	1.13437	21.9369	27.2058
TTTTTTTTCTT	371	346.918	375	367.597	1.02014	7.47726	24.8994
TTTTTTTTTGG	232	208.601	234	219.9	1.06412	14.5428	24.6648
TTTTTGTTGTT	140	117.58	142	123.53	1.14952	19.7871	24.4338
TTCTTTTTTTG	112	90.4012	113	94.8801	1.19098	19.7494	23.995
TTTTTTTTTTG	513	489.643	530	521.6	1.0161	8.46728	23.9056
TATTTTTTTTA	59	39.3935	59	41.2671	1.42971	21.0908	23.8322
TTTTTTCTTAC	50	31.682	50	33.1793	1.50696	20.5048	22.8136
ATTTTTTTTTA	64	44.9166	65	47.0625	1.38114	20.9892	22.6608
CATATATATAA	35	18.3256	35	19.1821	1.82462	21.0479	22.6468
TTTTATTTTTT	165	143.839	170	151.265	1.12386	19.85	22.6468
CTTCTTCTTTT	143	122.363	145	128.578	1.12772	17.4284	22.2866
TTTTTTTCTTT	369	347.908	378	368.66	1.02534	9.45759	21.7187
ATTTTTTTTTC	74	55.589	74	58.2679	1.27	17.6871	21.1698
TTATTTATTTT	109	90.2078	110	94.6765	1.16185	16.5016	20.6263
TTCTTTTTCTT	171	152.266	175	160.178	1.09254	15.4878	19.8418
ATTCTTTTTCA	25	11.4326	25	11.964	2.08961	18.4244	19.5601
TTAAAAAAAAA	104	86.2662	104	90.5263	1.14884	14.43	19.4432

Table A.34: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTC	74	51.6513	74	54.3831	1.36072	22.7929	26.6067
CTTCTTCTTTTT	99	76.131	101	80.2305	1.25887	23.2518	26.0038
TTTTTGTTGTTG	85	63.952	85	67.3652	1.26178	19.7644	24.1841
TTATATATATAA	26	10.9876	26	11.5513	2.25083	21.0938	22.3944
TTCTTTCTTTTT	91	71.2962	91	75.122	1.21136	17.4489	22.2054
TTTTTGTTTGTT	100	82.3399	100	86.7938	1.15216	14.1634	19.4315
TTTTTATTTTTT	83	65.7457	86	69.2593	1.24171	18.6182	19.3428
GATTTTTTTTTT	95	78.7701	96	83.0199	1.15635	13.9457	17.7977
TTTTTTTTTTAA	140	123.362	142	130.234	1.09034	12.2819	17.7122
AGTTTTTTTTTT	165	148.604	168	157.03	1.06986	11.345	17.2689
TTTTTTCTTTTC	83	67.7611	83	71.3878	1.16266	12.5093	16.8367
TTTTTCTTTCTT	107	91.7187	107	96.7138	1.10636	10.8148	16.489
TTTTTTTTCTTT	210	194.362	211	205.733	1.0256	5.33391	16.2505
TCTTTTTTTTTT	238	222.358	246	235.612	1.04409	10.6135	16.1799
GAAAAAAAAAAC	30	18.0617	30	18.9933	1.57951	13.7134	15.222
ATTTTTTTTTTC	40	27.5987	40	29.0324	1.37777	12.8187	14.8444
TTTTTTGTTTGT	77	63.7833	77	67.1871	1.14605	10.497	14.5002
GTATATATATAC	14	5.10958	14	5.37051	2.60683	13.4139	14.1112
GTTTTGATTTTG	22	11.717	22	12.3184	1.78594	12.7588	13.8599
TTTTTTTTTTCT	203	189.622	205	200.68	1.02153	4.36617	13.839
TCTTTTCTTTTT	81	68.3086	81	71.966	1.12553	9.57864	13.8034
CTTTTATTTTTC	23	12.75	23	13.4049	1.71579	12.417	13.5691
CTTTTCTTTTTT	58	45.9915	58	48.4138	1.19801	10.4782	13.4552
CAAAAAAAAAAA	105	92.4668	106	97.5053	1.08712	8.85444	13.3467
TTTTTTTTTGTA	84	71.7453	84	75.5963	1.11117	8.85436	13.2464

Table A.35: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTGG	116	93.3037	116	98.8491	1.17351	18.5596	25.2567
GTTTTTTTTTGT	53	37.4636	53	39.6075	1.33813	15.4374	18.3868
TTTTTTTTTTCA	55	39.7934	56	42.0743	1.33098	16.0113	17.7998
CTTTTTTTTTTG	52	37.8771	52	40.0453	1.29853	13.584	16.4786
TTTTTTTCTTTT	117	101.657	117	107.733	1.08602	9.65482	16.4463
ATTTTTTTTTTA	25	13.0028	25	13.7344	1.82025	14.9743	16.3427
CTTCTTCTTTTT	67	52.7121	67	55.7604	1.20157	12.3031	16.0697
TTTTTTTTTCTT	143	128.458	144	136.272	1.05671	7.94344	15.3356
TTTCTTCTTCTTT	41	28.5312	41	30.1538	1.35969	12.5976	14.8656
TGTTTTTGTTTTT	50	37.6867	50	39.8437	1.2549	11.3529	14.1358
ATTTTTTTGTTTG	17	7.81057	17	8.24841	2.061	12.2943	13.2215
ATTGTTGTTGTTA	10	2.706	10	2.85714	3.5	12.5276	13.0712
TTTTTTTTTTGTG	49	38.2011	49	40.3883	1.21322	9.47067	12.1988
GTGTTTGTTTGTG	9	2.33992	9	2.47059	3.64286	11.6349	12.124
ATTTTTTTTTTC	23	13.5966	23	14.3618	1.60147	10.8311	12.0906
TTTTTTGTTTTTT	60	49.1506	61	51.986	1.17339	9.75385	11.9673
TTTTTTCTTTTTC	45	34.5281	45	36.5	1.23288	9.42076	11.92
CTTTTTCTTTTTT	35	25.0969	36	26.5208	1.35742	11.0012	11.6412
TTTTTTTTTCTTT	133	122.022	133	129.413	1.02771	3.63592	11.4573
CTGTTTTTTTTTT	62	51.5472	62	54.5258	1.13708	7.96458	11.4474
TTTTGTTGTTGTT	64	53.8158	64	56.9302	1.12418	7.49162	11.0922
CTTTTTTTTTAAC	11	4.09054	11	4.31925	2.54674	10.283	10.8814
GTTTTTTTCTTC	17	9.03192	17	9.53867	1.78222	9.82361	10.7516
TTGTTTTTTTTC	25	16.2999	25	17.2191	1.45188	9.32146	10.6929
CTTTGTTTTTTT	27	18.2221	27	19.2511	1.40252	9.1333	10.6164

Table A.36: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTTG	38	25.2591	38	26.8172	1.417	13.2447	15.5191
ATTTTTTTTTTTA	15	6.60565	15	7.0082	2.14035	11.4145	12.3019
TTTTTCTTTCTTTC	36	26.0917	36	27.7019	1.29955	9.43262	11.5885
TTTTTTTTCTTTTT	60	49.6315	60	52.7409	1.13764	7.73716	11.3831
TTATATATATATAA	8	1.95496	8	2.07373	3.85778	10.8007	11.2726
TTTTTTTTTTTGG	80	70.0292	80	74.4734	1.07421	5.7268	10.6492
TTTTTTTTTCTTT	88	78.2763	88	83.2696	1.05681	4.86233	10.3041
CTTCTTCTTTTTT	49	39.8669	49	42.3491	1.15705	7.14783	10.1074
CTTCTTTTTTTTC	15	7.89892	15	8.38068	1.78983	8.73181	9.61987
TTTTTTTTTGGTT	51	42.3284	51	44.968	1.13414	6.4196	9.50471
GTTTTCTTTTCTTC	7	1.85455	7	1.96721	3.55833	8.88505	9.29789
GTTTTTTTTTTAT	12	5.66513	12	6.01015	1.99662	8.29748	9.00692
TTTTTTATTTTTT	25	17.5069	26	18.5814	1.39925	8.73433	8.90695
GTTTTTTTTGTTT	23	15.9593	23	16.9378	1.35791	7.03677	8.40539
CTTTTTTTTTTTT	122	113.925	122	121.354	1.00532	0.647364	8.35467
CTTTGTTTTTCTTC	8	3.05251	8	3.2381	2.47059	7.23565	7.70783
CTTTTCTTTTTCTA	7	2.37198	7	2.51613	2.78205	7.16232	7.5753
TTTTTTGTTTTTTT	36	29.2106	37	31.0169	1.1929	6.52621	7.52351
TCTTTTTTTTTTTC	21	14.7974	21	15.704	1.33724	6.10279	7.35148
CTTTTTTTTTTTGT	27	20.5806	27	21.8462	1.23592	5.71892	7.33024
GTTTTTTTCTTTTA	7	2.47155	7	2.62176	2.66996	6.87445	7.28745
GTTTTCTTTCTTTT	10	4.83094	10	5.125	1.95122	6.68455	7.27543
GTTTTTGTTCTTTC	5	1.17844	5	1.25	4	6.93147	7.22623
TTTTTTTTTGTTTT	62	55.2193	64	58.6911	1.09045	5.54208	7.18093
TTTTTTTTTTTGAA	22	15.9571	22	16.9355	1.29905	5.75589	7.065

Table A.37: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTT	96	82.207	96	87.8749	1.09246	8.48967	14.8904
CATATATATATATAC	7	1.17621	7	1.25348	5.58444	12.0399	12.4853
TTTTTTTTTTTTTTA	62	51.9437	62	55.462	1.11788	6.90903	10.9723
CTCTTTTTTTTTTTT	28	19.5792	28	20.88	1.341	8.21556	10.0166
GTTTTTTTTTTTTTT	96	87.5799	96	93.6372	1.02523	2.39239	8.8125
TTTTTTCTTTTTTTT	19	12.5928	19	13.4259	1.41517	6.59778	7.8149
TTTTTTTTTTTTTTC	66	58.6749	66	62.6649	1.05322	3.42234	7.76447
TTTTTTTTTTTTTGG	56	49.0347	56	52.3502	1.06972	3.77414	7.4381
TTTTTTTTTTCCTTT	16	10.0716	16	10.7368	1.4902	6.38252	7.40596
GTCTCTCTCTCTCTG	4	0.669023	4	0.712963	5.61039	6.89848	7.15293
TAGAAGAAGAAGAAT	3	0.278728	3	0.29703	10.1	6.93761	7.1284
ATTGTTTTTTTTTA	7	2.53342	7	2.7	2.59259	6.66861	7.11438
TTTTATTTTTATTTT	19	13.2218	19	14.0968	1.34783	5.67137	6.88894
GTCTTTTTTTTTTTG	10	5.06636	10	5.4	1.85185	6.16186	6.79963
TGTTTTTTTTTTTTG	22	16.3639	22	17.449	1.26082	5.09875	6.51116
TTTTTTTTTTGGTTT	36	30.123	36	32.137	1.1202	4.08641	6.41624
AGTTTTTTTTTTTTT	47	41.0154	47	43.7755	1.07366	3.34043	6.40138
GTTTGTTGTTGTTGC	3	0.360916	3	0.384615	7.8	6.16237	6.35317
TTTTTTTTTTTTGTG	20	14.5966	20	15.5634	1.28507	5.01623	6.29892
TCTTTTTTTTTTTTG	16	10.9125	16	11.6337	1.37531	5.09889	6.12283
CTTCTTCTTTTTTTT	26	20.5727	26	21.9403	1.18503	4.41406	6.08739
ATTTTTTTCTTTCTC	4	0.875805	4	0.933333	4.28571	5.82115	6.07563
TTTTCTTTTTCTTTT	27	21.6241	27	23.0625	1.17073	4.25598	5.99473
CACTCTTTTTTTTTG	4	0.93836	4	1	4	5.54518	5.79966
TTTTTTTTTTTTAAT	32	26.8192	32	28.6087	1.11854	3.58481	5.65179

Table A.38: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTTT	77	63.8121	77	68.4864	1.12431	9.02216	14.4654
TTTTTTTTTTTTTTTG	76	66.4643	76	71.34	1.06532	4.80904	10.1892
TTTTTTTTTTTTTTTC	51	43.9038	51	47.0844	1.08316	4.07411	7.64109
TTTTTTTTTTTTTTGT	41	34.1586	41	36.6197	1.11962	4.63239	7.48491
TCTTTTTTTTTTTTTT	49	42.5585	49	45.6393	1.07364	3.48146	6.90607
GTTTTTTTTTTTTTTA	12	6.89538	12	7.38462	1.625	5.82609	6.64865
TTTTTTATTTTTTTTT	13	7.80322	13	8.35714	1.55556	5.74383	6.63537
ATTTTTTTTTTTTTTT	31	25.2957	31	27.1092	1.14352	4.15756	6.30399
GCATATATATATATAC	5	1.56901	5	1.68	2.97619	5.45322	5.79495
TTCTTTTTTTTTTTGG	8	3.92219	8	4.2	1.90476	5.15486	5.70233
TGTTTTTTTTTTTTTT	41	35.817	41	38.4	1.06771	2.6861	5.54118
GTTATATATATATATT	2	0.149438	2	0.16	12.5	5.05146	5.18804
GTTCTTTTTTTTTGT	4	1.09877	4	1.17647	3.4	4.8951	5.16841
ATGTTTTTTTTTTTT	19	14.5623	19	15.6	1.21795	3.74619	5.05406
ATTTGTTTTTGTTTTA	2	0.169816	2	0.181818	11	4.79579	4.93238
GTTTTTTTTTTAATTA	5	1.86785	5	2	2.5	4.58145	4.92324
TTTTTTTTTTGTTTTT	21	16.6263	21	17.8125	1.17895	3.45706	4.90424
TTTGTTGTTGTTGTTT	7	3.474	7	3.72	1.88172	4.42531	4.90423
GCTCTGTTTTTTTCTC	5	1.94568	5	2.08333	2.4	4.37734	4.71914
ACTCTTTTTTTTTTC	3	0.622647	3	0.666667	4.5	4.51223	4.71716
ACAAAAAAAAAAAAG	6	2.73653	6	2.93023	2.04762	4.30007	4.71041
TTCTGTTTTTTTTTT	16	11.9731	16	12.825	1.24756	3.53908	4.63888
CTTTTCTTTTCTTTTT	13	9.11517	13	9.76271	1.3316	3.72293	4.61512
TTTTTTTTCTTTTTTT	16	12.003	16	12.8571	1.24444	3.49903	4.59885
GTTTTTTTTCTTTTTA	3	0.653779	3	0.7	4.28571	4.36586	4.5708

Table A.39: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTTT	60	50.4444	60	54.3687	1.10358	5.91336	10.4083
TTTTTTTTTTTTTTTG	57	49.7906	57	53.6626	1.06219	3.43908	7.70788
CTTTTTTCTTTTCTTC	4	0.929559	4	1	4	5.54518	5.83736
TTTTTTTTTTTTTGGA	10	5.64275	10	6.07143	1.64706	4.98991	5.72214
ATTCTTTTTTTTTTTG	4	1.0495	4	1.12903	3.54286	5.05973	5.35193
ATTTTCTTTTTTTTTG	3	0.536292	3	0.576923	5.2	4.94598	5.16506
TTTTTTTTTTTTTTTC	38	33.4327	38	36.0104	1.05525	2.04354	4.86591
TCTTTTTTTTTTTTTT	41	36.4846	41	39.3021	1.0432	1.73408	4.78399
ATTTTTTATTTTTTTA	3	0.619714	3	0.666667	4.5	4.51223	4.73133
TGTTTTTTTTTTTTTT	28	23.7994	28	25.625	1.09268	2.48181	4.55126
CTTTTTTCTTTTTTC	2	0.206574	2	0.222222	9	4.39445	4.54048
TTTTTTTTTTTGGTTT	22	18.0075	22	19.3846	1.13492	2.78438	4.40556
TTTTTTTTTTTTTTGG	28	23.9989	28	25.84	1.08359	2.24786	4.31752
GCTGTTTTTTTTTTTA	2	0.232396	2	0.25	8	4.15888	4.30492
CTTTTCTTTTCTTTTT	6	2.93838	6	3.16129	1.89796	3.84468	4.2834
TTTTTTATTTTATTTTT	5	2.12461	5	2.28571	2.1875	3.9138	4.27925
ATATATATATATATATT	25	21.0749	25	22.6891	1.10185	2.42474	4.26988
GTATATATATATATC	2	0.243711	2	0.262172	7.62857	4.0638	4.20984
TAAACAAAAAAAAAAG	3	0.796769	3	0.857143	3.5	3.75829	3.97741
GCTTTTTTTTTTCTTA	3	0.796769	3	0.857143	3.5	3.75829	3.97741
GTCTGTTTTTTTTTTG	2	0.275432	2	0.296296	6.75	3.81909	3.96512
CTTCTTTTCTTTTTCTT	4	1.48726	4	1.6	2.5	3.66516	3.95743
CTGTTTTTTTTTTTTG	4	1.50156	4	1.61538	2.47619	3.62689	3.91915
CATTTTTTTTTTTTTA	2	0.293552	2	0.315789	6.33333	3.69165	3.83769
GTTTTTTTTTTTTTGT	8	4.9538	8	5.33	1.50094	3.24872	3.8343

Table A.40: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTCTTCTTCTTCTTCTC	3	0.269144	3	0.290909	10.3125	7.00007	7.23336
CTTTTTTTTTTTTTTTTG	12	7.79118	12	8.42365	1.42456	4.24637	5.18297
TTTTTTTTTTTTTTTTTC	29	24.397	29	26.3941	1.09873	2.73052	5.01222
CTATATATATATATATAA	2	0.182431	2	0.197183	10.1429	4.63354	4.78906
CTTTTTTTTTTTTTTTGA	4	1.26599	4	1.36842	2.92308	4.29055	4.60176
GTTTTTTTTTTTTTTTAT	3	0.676952	3	0.731707	4.1	4.23296	4.4663
TTTTTTTTTTTTTTTTAA	16	12.4278	16	13.439	1.19056	2.79082	4.04243
CCTTTTTTTTTTTTTTTC	3	0.780983	3	0.844156	3.55385	3.80409	4.03744
ACTTTTTTTTTTTTTTTT	15	11.5298	15	12.4675	1.20313	2.77384	3.94671
TTTTTTTTTTTTTTTTGG	23	19.4147	23	21	1.09524	2.09235	3.89763
AAGTTTTTTTTTTTTTT	9	5.88627	9	6.36364	1.41429	3.11962	3.82142
AAAGAAGAAGAAGAAGAT	3	0.87074	3	0.941176	3.1875	3.47771	3.71107
CATATATATATATATATG	3	0.92913	3	1.00429	2.98718	3.28299	3.51636
GTCTTCTTCTTCTTTTTG	2	0.370071	2	0.4	5	3.21888	3.37441
GTTTTTTTTTTTTTTGT	5	2.5562	5	2.76316	1.80952	2.96532	3.35458
GTTTTTTTTTTAATTTC	3	1.00926	3	1.09091	2.75	3.0348	3.26818
GAGAAGAAGAAGAAGAAA	3	1.02795	3	1.11111	2.7	2.97976	3.21314
ACTCTCTCTCTCTCTCTA	1	0.0405193	1	0.0437956	22.8333	3.12822	3.20598
GTTTTTTTTTTTTTTTT	30	26.9625	30	29.1724	1.02837	0.839216	3.20252
CTCTTTTTTTTTTCTTC	2	0.41119	2	0.444444	4.5	3.00815	3.16369
AGTCTTCTTCTTCTTCTA	2	0.41119	2	0.444444	4.5	3.00815	3.16369
TTCTTTTTTTTTTTTTT	16	13.1538	16	14.2245	1.12482	1.88198	3.13403
CTTTTCTTTTTCTTTTTC	3	1.05732	3	1.14286	2.625	2.89524	3.12863
CAAAAAAAAAAAAAACT	3	1.06749	3	1.15385	2.6	2.86653	3.09992
CTTTCTTTTTCTTTTTTC	2	0.427005	2	0.461538	4.33333	2.93267	3.08822

Table A.41: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTTTTTTTTTG	7	2.03734	7	2.21277	3.16346	8.06167	8.63987
TTTTTTTTTTTTTTTTGG	19	14.4824	19	15.7368	1.20736	3.58025	5.15864
TTTTTTTTTTTTTTTTTC	23	18.4518	23	20.0532	1.14695	3.15344	5.06759
CGAAGAAGAAGAAGAAGAT	2	0.171309	2	0.186047	10.75	4.74981	4.91487
GAAAAAAAAAAAAAAAAT	3	0.657691	3	0.714286	4.2	4.30525	4.5529
CCTTTTTTTTTTTTTTTT	9	5.52359	9	6	1.5	3.64919	4.39378
TAGAGAGAGAGAGAGAGAT	2	0.277972	2	0.301887	6.625	3.7817	3.94677
CTTGTTTTGTTTTGTTTTA	2	0.334829	2	0.363636	5.5	3.4095	3.57457
ATGTTTTTTTTTTTTTTG	4	1.64417	4	1.78571	2.24	3.2259	3.55624
GTGTTGTTGTTGTTGTTGA	1	0.0287747	1	0.03125	32	3.46574	3.54826
CTTTTTTTTTTTTTTTGC	3	0.969218	3	1.05263	2.85	3.14196	3.38963
TTTTCTTCTTCTTTTTTT	5	2.5575	5	2.77778	1.8	2.93893	3.35203
AGTTTTTTTTTTTTTTTA	4	1.75377	4	1.90476	2.1	2.96775	3.2981
GTCTCTCTCTCTCTCTCTT	2	0.406012	2	0.440945	4.53571	3.02397	3.18904
GTTTTTTTTTTTTTTTTT	22	19.0877	22	20.7447	1.06051	1.29256	3.12402
TTTCTTTCTTTCTTTTTTT	7	4.5111	7	4.9	1.42857	2.49672	3.07558
CAAAAAAAAAAAAAAAAG	3	1.09613	3	1.19048	2.52	2.77278	3.02047
ATATATATATATATATATT	19	16.2225	19	17.6289	1.07778	1.42312	3.00277
CAGAGAGAGAGAGAGAGAC	1	0.0521201	1	0.0566038	17.6667	2.87168	2.9542
CTGTTTTGTTTTGTTTTTT	2	0.460387	2	0.5	4	2.77259	2.93767
TTTGTTGTTGTTGTTGTTG	12	9.39696	12	10.209	1.17544	1.9397	2.93425
ATATATATATATATATATG	14	11.3578	14	12.3402	1.1345	1.76672	2.92808
GAGAAGAAGAAGAAGAAGT	1	0.0541641	1	0.0588235	17	2.83321	2.91574
TTATATATATATATATATA	13	10.4283	13	11.3299	1.14741	1.78756	2.8655
CGATGATGATGATGATGAG	1	0.0575493	1	0.0625	16	2.77259	2.85511

Table A.42: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTTTTTT	22	17.3717	22	18.9692	1.15977	3.26093	5.1964
GCTCTCTCTCTCTCTCTCTT	2	0.336286	2	0.366972	5.45	3.39123	3.56588
TTTTTTTTTTTTTTTTTTG	17	14.1565	17	15.4564	1.09987	1.61822	3.11172
TTTTTTTTCTTTTTTTTTT	7	4.48954	7	4.9	1.42857	2.49672	3.10911
CTATATATATATATATATAA	1	0.046695	1	0.0509554	19.625	2.9768	3.06412
TTCATCATCATCATCATCAC	3	1.09962	3	1.2	2.5	2.74887	3.01094
TCTTTTTTTTTTTTTTTTG	5	2.74888	5	3	1.66667	2.55413	2.99122
TACTTTTTTTTTTTTTTTC	2	0.488732	2	0.533333	3.75	2.64351	2.81817
ATTTTTTTTTTTTTTTTTA	2	0.488732	2	0.533333	3.75	2.64351	2.81817
CTGTTGTTGTTGTTGTTGTC	1	0.060091	1	0.0655738	15.25	2.72458	2.81189
TTTTTTTTTTTTTTTTGTA	6	3.78998	6	4.13636	1.45055	2.23165	2.7564
CTTTTTTTTTTTTGGTTTT	3	1.20271	3	1.3125	2.28571	2.48004	2.74211
AACTTTTTTTTTTTTTTTG	3	1.2218	3	1.33333	2.25	2.43279	2.69487
AGTTTTTTTTTTTTTTTTG	3	1.2218	3	1.33333	2.25	2.43279	2.69487
TTATATATATATATATATAG	2	0.531146	2	0.579618	3.45055	2.47707	2.65173
GTGTTGTTGTTGTTGTTGTA	1	0.0751138	1	0.0819672	12.2	2.50144	2.58875
CTTTTTTTTTTTTTTTTCAC	1	0.0763657	1	0.0833333	12	2.48491	2.57222
GCTTTTTTTTTTTTTTTTGA	1	0.0763657	1	0.0833333	12	2.48491	2.57222
CTTTTTTTTTTTTTTTCTTA	2	0.56392	2	0.615385	3.25	2.35731	2.53198
TCACACACACACACACACAA	1	0.083308	1	0.0909091	11	2.3979	2.48521
CCTCTCTCTCTCTCTCTCTG	1	0.0840723	1	0.0917431	10.9	2.38876	2.47608
TAGAGAGAGAGAGAGAGAGT	1	0.087275	1	0.0952381	10.5	2.35138	2.43869
GTTTTTTTTTTTTTTTTTA	3	1.34397	3	1.46667	2.04545	2.14686	2.40895
CTCTTCTTCTTCTTCTTCTC	1	0.0916387	1	0.1	10	2.30259	2.3899
CTTTTTTTCTTTTTTTTTA	1	0.0916387	1	0.1	10	2.30259	2.3899

A.3 The *Arabidopsis thaliana* Core Promoters

A.3.1 Timing Information

Table A.43: Full Size and Timing Information, Core Promoters

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	0.350471	0.722272	4.769494	1.072743
2	0	0.000001	0.624713	0.880534	8.525701	1.505247
3	1	0.000003	0.945494	0.607531	5.158450	1.553025
4	2	0.000012	1.423833	0.645800	1.885470	2.069633
5	3	0.000046	1.866457	1.027063	1.101381	2.89352
6	4	0.000186	2.408993	2.151728	0.807213	4.560721
7	5	0.000743	2.986029	3.723123	0.807073	6.709152
8	6	0.002962	3.704660	12.161700	1.098893	15.86636
9	7	0.011242	4.848222	49.126031	2.404455	53.974253
10	8	0.036175	6.160703	111.959462	5.726779	118.120165
11	9	0.091290	7.678349	230.995124	11.069305	238.673473
12	10	0.181564	9.273617	320.370791	16.753109	329.644408
13	11	0.299251	10.880403	349.471948	21.414850	360.352351
14	12	0.432477	12.564024	357.198369	25.302929	369.762393
15	13	0.572592	13.819888	413.397615	28.417219	427.217503
16	14	0.714941	15.266773	385.487291	32.630946	400.754064
17	15	0.857370	17.166468	403.000797	33.958973	420.167265
18	16	0.998946	18.556766	338.292306	36.425449	356.849072
19	17	1.139256	20.128509	383.146666	39.245750	403.275175
20	18	1.278112	21.795687	389.445697	42.441547	411.241384

A.3.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.44: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
G	27163	27169	410665	410665	1	0	-5.99725
C	27168	27169	515540	515540	1	0	-0.999961
T	27169	27169	874597	874597	1	-9.70998e-11	0
A	27169	27169	916050	916050	1	2.03404e-10	0

Table A.45: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AG	26631	27015.9	139244	138465	1.00562	780.957	-382.106
AC	26939	27130	150313	173826	0.864731	-21846	-190.296
CG	22793	25645.4	67028	77926.3	0.860146	-10098	-2687.56
CC	25045	26448.9	104141	97827	1.06454	6513.55	-1365.93
GC	24169	25645.4	66256	77926.3	0.85024	-10749.2	-1433.08
GA	26689	27015.9	144470	138465	1.04337	6133.13	-324.875
GG	22418	24412.5	72843	62073.9	1.17349	11653.4	-1910.7
CA	27044	27130	167783	173826	0.965234	-5936.98	-85.8329
CT	26906	27116	168498	165960	1.01529	2557	-209.197
GT	26672	26974.3	123674	132199	0.935511	-8244.41	-300.568
AT	27154	27168.7	246131	294891	0.834652	-44485.9	-14.6836
TC	26984	27116	190001	165960	1.14486	25703.5	-131.69
TG	26613	26974.3	127384	132199	0.963575	-4726.63	-358.838
AA	27144	27168.8	373343	308868	1.20875	70780.3	-24.8125
TA	27116	27168.7	220958	294891	0.749288	-63775.6	-52.6365
TT	27136	27168.5	327616	281546	1.16363	49648.7	-32.4425

Table A.46: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTG	13764	13012.5	22574	18013	1.25321	5095.11	772.838
GCC	11100	10423.7	16447	13384	1.22886	3389.5	697.78
GAG	15428	14904	29905	21960.1	1.36179	9234.57	533.056
GGC	9912	9403.77	14386	11752.4	1.22409	2908.86	521.727
CCA	19459	19229.4	37230	33892.8	1.09846	3496.36	230.942
TGG	15412	15184.1	25476	22595.1	1.1275	3057.17	229.651
CAC	17252	17155.4	34553	27531.2	1.25505	7849.55	96.9086
AGC	15192	15127.4	23104	22465.4	1.02843	647.599	64.7787
ACG	13802	13777.7	21610	19543	1.10577	2172.69	24.3202
TTG	22364	22383.4	51973	47716.9	1.0892	4440.53	-19.3608
TAT	23734	24051.9	74678	59368.6	1.25787	17132.6	-315.833
ATA	23941	24359.3	81342	62182.5	1.30812	21847.6	-414.709
ACA	22134	22590.1	55268	48919.5	1.12977	6743.67	-451.49
GCT	14272	14767.2	21069	21655	0.972941	-577.97	-486.795
TGT	19907	20426.3	41830	38362.4	1.09039	3619.82	-512.594
CAA	24409	24934.6	66001	68381.2	0.965192	-2338.3	-520.011
GCG	6609	7266.18	8543	8614.28	0.991725	-70.9867	-626.522
AAC	23627	24262.1	59689	61261.2	0.974336	-1551.83	-626.752
ACC	17485	18137.9	31513	30363.8	1.03785	1170.69	-640.993
ATT	25575	26247.1	82818	92198.4	0.898258	-8886.16	-663.377
TAA	25494	26170.2	83169	90053.1	0.923555	-6613.98	-667.39
AGA	21897	22601.2	58588	48985.4	1.19603	10487.7	-693.105
AAT	25680	26488.3	87870	100313	0.875962	-11636.8	-795.878
TTA	25025	25858.8	73005	82768.8	0.882035	-9163.86	-820.225
AAG	22901	23736.1	57493	56749.9	1.01309	747.914	-820.227

Table A.47: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ACTA	8377	7038.61	10517	8385.51	1.25419	2381.98	1458.26
ATTA	14354	13142.6	22572	18454.9	1.22309	4545.53	1265.62
TAGT	7751	6601.78	9557	7786	1.22746	1958.68	1243.9
GAAG	8200	7055.58	11616	8409.06	1.38137	3752.83	1232.58
CTTC	11172	10101.9	18478	12991	1.42237	6510.2	1124.88
GTCG	3574	2713.16	4178	2945.18	1.41859	1460.88	984.898
ATCA	11403	10502.6	16094	13653.2	1.17877	2647.01	937.939
TAAT	14580	13696	23214	19574.7	1.18592	3958.45	911.889
CGAC	3879	3117	4529	3411.02	1.32775	1283.92	848.364
CCTA	6053	5318.52	7014	6095.08	1.15076	984.944	783.004
TGAT	9034	8311.13	11810	10208.3	1.15691	1721.31	753.438
GCAG	2825	2164.28	3106	2324.11	1.33643	900.739	752.642
TACT	7701	7025.48	9468	8367.3	1.13155	1170.12	707.003
GATT	10186	9503.86	13385	12030.8	1.11256	1427.68	706.055
CATC	8009	7338.13	10425	8804.06	1.18411	1761.75	700.637
TTAG	8876	8203.77	11004	10049.8	1.09494	998.084	699.052
CCCA	7273	6606.98	9419	7793.06	1.20864	1784.86	698.512
GTAG	4215	3604.11	4790	3983.32	1.20251	883.347	659.964
TTAC	9041	8426.12	11258	10378.9	1.0847	915.296	636.792
GTCA	6779	6175.71	8031	7213.27	1.11336	862.421	631.837
AGTA	7604	7005.34	9390	8339.39	1.12598	1114.17	623.543
TGCA	5239	4654.4	6044	5258.2	1.14944	841.792	619.868
GCTT	7299	6726.45	8771	7955.8	1.10247	855.609	596.252
TCCT	8262	7708.79	10892	9330.77	1.16732	1685.11	572.603
AAGC	8391	7853.77	10347	9539.5	1.08465	840.75	555.197

Table A.48: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAAAT	8078	6840.11	9993	8195.87	1.21927	1981.15	1343.69
ATTCA	4264	3480.52	4747	3876.96	1.22441	961.083	865.697
TTATT	7234	6424.77	9518	7625.14	1.24824	2110.47	858.182
ATTTA	6630	5848.17	7911	6851.3	1.15467	1137.73	831.901
ACGTG	2042	1398.1	2255	1494.75	1.50861	927.235	773.542
ACTTG	2444	1798.28	2576	1937.39	1.32963	733.897	749.834
GCCCA	2540	1893.53	3383	2043.76	1.65528	1704.94	746.056
ACTCA	2711	2068.03	2929	2239.67	1.30778	785.944	733.926
ATAAA	11100	10414.6	15134	13646.8	1.10897	1565.4	707.442
TCTAT	4274	3629.03	4760	4054.68	1.17395	763.389	699.163
TGAAT	3338	2724.33	3645	2988.77	1.21957	723.514	678.111
CAAGT	2484	1931.5	2632	2086.27	1.26158	611.582	624.909
TAGAT	2656	2101.17	2864	2277.03	1.25778	656.846	622.369
TAATT	7025	6444.02	8568	7651.35	1.1198	969.492	606.412
CCCTA	1966	1449.67	2089	1551.41	1.34652	621.523	598.971
CACGT	2172	1652.47	2419	1775.32	1.36257	748.378	593.77
CTATA	4596	4046.1	4938	4559.8	1.08294	393.468	585.682
TACAT	3183	2648.53	3526	2901.24	1.21534	687.662	585.097
GATTC	2650	2126.4	2879	2305.5	1.24875	639.555	583.346
TATCT	4229	3689.96	4719	4127.92	1.14319	631.507	576.62
GAATC	2627	2118.93	2818	2297.07	1.22678	575.984	564.63
ATCTA	3065	2557.73	3345	2796.74	1.19603	598.791	554.541
TGGGC	1860	1384.2	2306	1479.49	1.55864	1023.44	549.551
TCATC	3811	3307.86	4587	3671.71	1.24928	1020.92	539.6
TCAAT	4433	3927.75	5005	4415.55	1.13349	627.151	536.441

Table A.49: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATAAA	5284	4351.25	5675	4987.01	1.13796	733.401	1026.26
TAAAAT	3421	2726.79	3822	3023.08	1.26427	896.25	775.914
CAAAAC	2740	2089.77	2984	2287.99	1.3042	792.52	742.282
AATATT	2664	2057.81	2869	2251.6	1.2742	695.217	687.81
TAAATA	3783	3185.49	4195	3564.24	1.17697	683.544	650.336
ATAAAT	4271	3692	4782	4173.82	1.14571	650.486	622.2
GTTTTG	1786	1282.87	1908	1382.96	1.37965	614.046	590.943
ATTTTA	2691	2181.32	2901	2392.5	1.21254	559.069	565.058
AATATC	1481	1079.37	1527	1159.11	1.31739	420.917	468.503
GATATT	1113	764.516	1146	816.17	1.40412	388.964	418.011
AAACCC	2106	1741.49	2268	1893.87	1.19755	408.861	400.244
TTATAA	2693	2321.57	2839	2553.32	1.11189	301.096	399.674
CAAATC	1657	1309.98	1726	1412.91	1.22159	345.472	389.393
ATTTTC	2467	2108.38	2615	2309.21	1.13242	325.196	387.521
TAAACC	1861	1522.62	2029	1648.92	1.2305	420.859	373.466
ACCCTA	1087	773.228	1138	825.604	1.37838	365.198	370.236
AGGCCC	960	657.769	1066	700.816	1.52108	447.105	362.957
TTAAAA	4110	3763.62	4525	4261.06	1.06194	271.949	361.852
TGAAAT	1510	1188.96	1578	1279.44	1.23335	330.958	360.938
AGAAAA	4445	4100.27	5097	4674.72	1.09033	440.806	358.833
ACCAAA	2707	2375.93	2920	2615.9	1.11625	321.133	353.136
TATTTA	2574	2251.99	2817	2473.42	1.13891	366.411	344.011
ATCTCA	1282	982.015	1329	1052.64	1.26254	309.825	341.743
CAAATT	2194	1880.93	2294	2051.02	1.11847	256.838	337.794
CACGTG	980	695.326	1074	741.349	1.44871	398.104	336.309

Table A.50: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATAAAT	2249	1634.88	2298	1793.17	1.28153	570.027	717.239
ATAAATA	2315	1823.62	2478	2007.45	1.2344	521.828	552.327
CAAAAAC	1149	723.764	1187	780.29	1.52123	497.968	531.047
TAAAAAT	1336	936.698	1405	1013.88	1.38577	458.385	474.379
CTATAAA	1271	960.099	1287	1039.67	1.2379	274.665	356.545
AATAAAA	2343	2018.53	2694	2230.4	1.20785	508.746	349.259
TAAATAC	852	568.001	862	610.593	1.41174	297.239	345.455
CTATATA	1560	1257.93	1587	1369.85	1.15852	233.515	335.735
ATTTTTA	955	673.608	988	725.54	1.36174	305.061	333.355
CTTTTTC	866	593.381	898	638.176	1.40714	306.718	327.39
TTTCTTT	1751	1479.08	1974	1617.45	1.22044	393.239	295.515
AAAAACA	1986	1712.52	2107	1881.12	1.12008	238.926	294.243
GAAAAAG	900	650.225	934	700.05	1.33419	269.296	292.569
GTTTTTG	596	383.013	615	410.329	1.4988	248.868	263.534
AGAAAAA	2224	1991.81	2385	2199.74	1.08422	192.848	245.228
TTATATT	728	522.556	755	561.267	1.34517	223.873	241.383
AACCCTA	876	667.888	915	719.302	1.27207	220.188	237.612
TTTTCTT	2035	1810.94	2220	1993.01	1.1139	239.457	237.386
CATCTTC	614	418.004	631	448.104	1.40816	215.98	236.086
TAAAAAG	1071	861.618	1099	931.303	1.18007	181.963	232.981
CAAAATC	812	610.887	842	657.217	1.28116	208.619	231.086
TTCTCTT	1254	1045.18	1337	1133.61	1.17942	220.633	228.413
ACAAAAA	2100	1886.92	2248	2079.68	1.08094	174.959	224.682
CTCTTTC	894	695.433	938	749.352	1.25175	210.62	224.547
TGTTTTT	1181	977.24	1243	1058.57	1.17423	199.639	223.663

Table A.51: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATAAATA	1373	1085.42	1387	1190.81	1.16476	211.533	322.7
CTATAAAT	726	480.418	730	521.15	1.40075	246.015	299.76
CTATATAA	640	414.007	642	448.558	1.43125	230.189	278.774
ATATAAAC	562	351.539	562	380.439	1.47724	219.28	263.68
TAAAAAAT	477	298.661	484	322.899	1.49892	195.896	223.335
ATATATAC	546	395.316	561	428.16	1.31026	151.596	176.321
AATATATT	301	183.13	301	197.573	1.52348	126.721	149.571
AAGAAAAA	1269	1130.99	1325	1241.87	1.06694	85.8567	146.112
TTATATAA	523	396.655	527	429.621	1.22666	107.665	144.617
TATATAAA	1267	1136.2	1283	1247.71	1.02828	35.7801	138.055
AGAAAAAA	1132	1002.36	1175	1097.97	1.07016	79.6735	137.684
TTTTAAAA	703	578.176	711	628.332	1.13157	87.8821	137.421
ATATAAAG	379	263.984	381	285.227	1.33578	110.305	137.064
CTCTTCTC	410	295.717	436	319.7	1.36378	135.274	133.969
ATTTTTTA	311	204.733	314	220.968	1.42102	110.332	130.024
TTATAAAA	512	400.23	513	433.521	1.18333	86.3565	126.098
ATAAATAC	582	469.484	586	509.186	1.15086	82.3366	125.034
ACAAAAAA	969	851.7	1000	930.308	1.07491	72.2395	125.03
AAATTAAA	727	616.837	754	670.827	1.12399	88.1283	119.462
GCCCATTA	378	277.969	401	300.414	1.33482	115.809	116.191
TTAAAAAA	819	711.213	842	774.821	1.0867	70.0105	115.571
AAAAAACA	901	796.46	933	869.072	1.07356	66.2233	111.119
GCCCAATA	324	230.908	342	249.338	1.37163	108.073	109.746
ATTAAAAA	714	615.79	725	669.675	1.08261	57.5497	105.656
AATAAAAA	1104	1010.3	1177	1106.83	1.0634	72.3525	97.9208

Table A.52: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAAAAAAAT	254	127.424	257	138.826	1.85123	158.274	175.213
TATATAAAC	404	284.571	404	310.93	1.29933	105.787	141.574
AATAAATAA	333	227.993	364	248.852	1.46272	138.427	126.15
TATAAATAC	401	300.108	402	327.999	1.22561	81.7833	116.217
AAGAAAAAA	694	594.038	713	652.778	1.09226	62.9184	107.937
AAAAAAAGA	628	532.323	651	584.291	1.11417	70.3794	103.802
CTATAAATA	489	402.38	490	440.605	1.11211	52.0659	95.3376
TTATATATT	164	91.9113	164	100.071	1.63884	81.0138	94.9629
AGAAAAAAA	692	605.959	710	666.024	1.06603	45.3965	91.8796
AAAGAAAAA	691	605.483	715	665.495	1.07439	51.3017	91.2904
TTCTCTCTT	247	172.158	253	187.717	1.34777	75.5086	89.1606
CTCTTTCTC	258	182.792	267	199.35	1.33935	78.0132	88.9099
AATAAAATA	415	338.145	450	369.83	1.21678	88.2923	84.9936
ATTTTTTTA	156	90.5285	156	98.5629	1.58275	71.6291	84.8938
AATATATAA	172	106.272	175	115.737	1.51204	72.356	82.8162
TTTTTTCTT	485	411.035	503	450.153	1.1174	55.8344	80.2537
TTTTATTTT	438	367.461	490	402.11	1.21857	96.8636	76.9135
AATAAAAAA	466	396.69	477	434.328	1.09825	44.7025	75.0403
AAGAGAGAA	162	103.507	162	112.72	1.43719	58.7558	72.5691
TTTTTAAAA	305	241.901	305	264.1	1.15487	43.9153	70.6943
TTTTTTGTT	349	285.324	352	311.756	1.12909	42.7365	70.3054
ATTAAAAAA	351	287.972	358	314.665	1.13772	46.1908	69.4711
ACAAAAAAA	596	530.476	614	582.244	1.05454	32.6064	69.4141
GCCCATTTA	117	65.841	120	71.6521	1.67476	61.8803	67.267
ATTTTTTTC	155	100.594	155	109.542	1.41498	53.803	67.0112

Table A.53: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTATATATAC	122	67.7584	122	74.5516	1.63645	60.0886	71.7449
TAAAAAAAAT	96	50.3785	97	55.4117	1.75053	54.3122	61.8993
CTATAAATAC	181	128.934	181	142.019	1.27448	43.8994	61.3949
TATAAATAAG	99	54.9764	99	60.474	1.63707	48.7977	58.2334
AAAAAAAAAG	514	461.754	528	511.735	1.03178	16.521	55.0959
CTATATAAAC	155	108.958	155	119.972	1.29197	39.7059	54.6317
CAAAAAAAAA	543	491.103	551	544.555	1.01183	6.48255	54.5472
AAAAAAAACT	216	168.514	217	185.75	1.16824	33.7426	53.6244
AAAGAAAAAA	394	347.902	402	384.751	1.04483	17.6301	49.0254
AAAAAAAAAC	495	449.166	503	497.669	1.01071	5.35971	48.0963
TTATATATAA	96	58.5304	96	64.3877	1.49097	38.3449	47.501
AGTTTTTTTT	151	111.772	151	123.077	1.22688	30.875	45.4234
AAAATAATAA	173	133.586	173	147.156	1.17562	27.991	44.7279
TCTCTCTTTC	208	168.33	212	185.547	1.14257	28.2547	44.0148
CTCTCTCTTT	222	182.339	224	201.04	1.1142	24.2236	43.6917
AAAAGAAAAA	374	332.811	386	367.959	1.04903	18.4757	43.6386
CTATATATAA	120	83.8348	120	92.2668	1.30058	31.5368	43.0373
ACAAAAAAAA	389	349.351	395	386.364	1.02235	8.73203	41.8183
TTTTTTTTGT	252	213.49	254	235.52	1.07846	19.1868	41.7919
TTCTCTCTCA	67	36.4125	67	40.0402	1.67332	34.4921	40.8553
AAGAAAAAAA	428	389.275	438	430.834	1.01663	7.22522	40.5903
AATAAATAAA	212	175.525	237	193.503	1.22478	48.0553	40.0264
AAAAAAAGAA	343	305.754	352	337.876	1.0418	14.4147	39.4281
TTTTTAAAAA	150	115.694	150	127.405	1.17735	24.4896	38.9524
CTATAAATAG	116	83.1387	116	91.4996	1.26776	27.5216	38.6372

Table A.54: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAA	383	334.293	388	373.714	1.03823	14.5555	52.0948
TAAAAAAAAAT	56	25.6148	56	28.4741	1.9667	37.876	43.8022
AAAGAAAAAAA	261	221.36	263	246.951	1.06499	16.5597	42.995
AAGAAAAAAAA	257	223.943	259	249.845	1.03664	9.32065	35.3846
ATTTTTTTTTA	47	22.2977	47	24.7853	1.89629	30.0752	35.0461
GAAAAAAAAAC	73	45.4996	73	50.597	1.44277	26.7595	34.5111
TTTTTTTTTAA	129	99.6401	130	110.912	1.1721	20.6433	33.3139
CTATATATATG	45	21.6347	45	24.048	1.87126	28.1975	32.9563
TCTCTCTCTTT	165	136.632	166	152.191	1.09073	14.4169	31.1284
TCTTTTTTTTT	195	166.599	198	185.674	1.06639	12.7266	30.6943
TTAAAAAAAAA	161	133.525	163	148.722	1.096	14.9424	30.1262
AATAAATAAAA	114	88.1673	114	98.121	1.16183	17.0996	29.2937
AAAAAGAAGAA	114	88.6957	114	98.71	1.1549	16.4173	28.6125
CAAAAACAAAC	45	23.9994	45	26.6776	1.68681	23.5278	28.2886
GAAAAAAAAAG	70	47.7591	71	53.1117	1.3368	20.61	26.7628
CTCTTTCTCTC	104	80.5327	105	89.6121	1.17172	16.6394	26.5956
AAAAATAAAAA	178	153.614	195	171.161	1.13928	25.4275	26.2269
AAAAAAAAACT	156	131.898	156	146.906	1.06191	9.37007	26.1813
GGATTTTATTG	27	10.5015	27	11.6706	2.31351	22.6467	25.4965
AAATAAAATAA	195	171.113	213	190.72	1.11682	23.5335	25.4815
TCTATATATAT	132	109.012	132	121.365	1.08763	11.0878	25.2575
CTTTCTCTCTC	92	70.4361	94	78.3627	1.19955	17.103	24.5716
ATTTTTTTTTG	50	30.93	50	34.3859	1.45408	18.7188	24.0149
TATAAATAAAG	37	19.4148	37	21.5796	1.71458	19.9492	23.8606
AAAAAAAAGAG	98	76.9934	100	85.6682	1.16729	15.4688	23.6422

Table A.55: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAA	273	240.103	277	270.964	1.02228	6.10324	35.0545
GTTTTTTTTTTC	37	16.3496	37	18.3758	2.01352	25.8958	30.2185
TCAAAAACAAAC	33	14.1181	33	15.8671	2.07978	24.1647	28.0187
AGAAAAAAAAAA	162	137.513	163	154.896	1.05232	8.31222	26.5487
GAAAAAAAAAAT	31	14.2393	31	16.0033	1.93709	20.4969	24.1174
TTAAAAAAAAAA	110	88.8072	110	99.9447	1.10061	10.5449	23.5414
AAAAAAAAAAGA	178	156.89	180	176.786	1.01818	3.24306	22.4703
CCAAAAAAAAAA	123	102.589	124	115.485	1.07374	8.82193	22.3182
GAAAAAAAAAAG	44	26.5588	45	29.8557	1.50725	18.4629	22.2125
AATAAAATAAAA	125	104.908	131	118.099	1.10924	13.581	21.9041
TAAAAAAAAAAG	41	24.0797	41	27.0676	1.51472	17.0246	21.8204
TCAAAAAAAAAA	60	41.9579	60	47.1797	1.27173	14.4229	21.4606
CTTTTTTTTTTC	36	19.9468	36	22.4202	1.6057	17.0481	21.2563
TTTTTTTTTTAA	88	69.4237	88	78.1028	1.12672	10.4994	20.8655
CTATATATATAC	26	11.6924	26	13.1403	1.97864	17.7427	20.7781
TTATATATATAG	19	6.44267	19	7.23979	2.62438	18.3321	20.5484
AAAAAAAAAACT	109	90.6621	109	102.036	1.06825	7.19665	20.0787
CACAAAAAAAAA	62	44.8982	62	50.4886	1.228	12.734	20.0096
CTCTCTCTCTTT	106	87.7754	106	98.7817	1.07307	7.47584	19.9978
GAAAAAAAAAAC	41	25.1817	41	28.307	1.44841	15.189	19.9856
TAAAAAAAAAAT	27	12.9099	27	14.5089	1.86093	16.7691	19.9218
CAAAACAAAAAA	52	36.0421	52	40.5233	1.28321	12.9671	19.0609
TTATATATATAA	22	9.31338	22	10.4662	2.102	16.3436	18.911
AAAAAATAAAAA	90	73.1265	94	82.274	1.14252	12.5246	18.6857
TTTTTTTTCTTT	102	85.2815	104	95.9708	1.08366	8.3561	18.2595

Table A.56: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAA	203	178.441	206	203.434	1.01261	2.58179	26.1768
AAAAAAAAAAACT	82	62.3211	82	70.8997	1.15656	11.9272	22.5024
AAAAAAAAAAAAC	158	137.959	160	157.166	1.01803	2.85953	21.4311
ATTTTTTTTTTTA	17	5.38436	17	6.11917	2.77815	17.3704	19.5452
AAAAAAGAAAAAA	66	49.2531	67	56.0195	1.19601	11.9925	19.3171
AATATATATATAA	14	3.57106	14	4.05827	3.44974	17.3362	19.1267
AAAACAAACAAAG	30	16.2406	30	18.4607	1.62508	14.5666	18.4104
CTTTTTTTTTTTT	125	108.351	127	123.369	1.02943	3.68408	17.8676
AAAAAAGAAGAAG	44	29.6493	44	33.7105	1.30523	11.7207	17.369
TCTTTTTTTTTTT	94	78.5012	94	89.3333	1.05224	4.78649	16.9371
AAAGAAAAAAAAA	102	86.6266	102	98.5946	1.03454	3.46355	16.6633
TTTTTTTTTTTCA	26	14.169	26	16.1053	1.61438	12.4527	15.783
CCAAAAAAAAAAA	92	77.7925	93	88.5258	1.05054	4.58544	15.4324
TTTTTTTTTTCTT	69	55.3354	70	62.9444	1.11209	7.43699	15.2278
AAAAAGAAAAAAA	81	67.4844	81	76.781	1.05495	4.3329	14.7868
ATCAAAAACAAAC	30	18.5603	30	21.0984	1.42191	10.5601	14.4051
TTTTTTTTTTTTG	107	93.7165	107	106.678	1.00302	0.322723	14.1834
TCAAAAAAAAAAA	50	37.6691	50	42.8351	1.16727	7.73331	14.1592
AAATAAAATAAAA	103	89.7837	107	102.194	1.04703	4.91757	14.1446
ATCATCTTCTTCA	16	6.76016	16	7.68293	2.08254	11.7374	13.7847
AAAAAAAAAAAGA	112	99.3338	112	113.084	0.990418	-1.07837	13.4415
TTTTTTTTTTGG	43	31.5767	43	35.9032	1.19766	7.75604	13.2775
ACAAAAAAAAAAC	23	12.9792	23	14.7526	1.55905	10.2138	13.1593
ATTAAAAAAAAAA	61	49.2467	61	56.0123	1.08905	5.20349	13.0559
TCTCTCTCTCTTT	87	74.9633	87	85.3018	1.01991	1.71502	12.9551

Table A.57: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAA	157	132.574	158	152.752	1.03436	5.33718	26.5498
AAAAAAGAAAAAAA	55	40.6725	55	46.7845	1.1756	8.89799	16.598
AAAAAAAAAAAAAG	110	95.3119	111	109.744	1.01144	1.26303	15.7658
AAAAAAAAAAAAAT	59	46.4093	59	53.389	1.1051	5.89599	14.1622
AAAAAAAAAAAAGA	79	66.0725	79	76.0367	1.03897	3.02031	14.1169
AAAAAAAAAAAAAC	116	103.025	118	118.642	0.994586	-0.640556	13.7593
AAAAAAAAAAAACT	65	53.2867	65	61.3084	1.06021	3.80056	12.9155
AAAAACAAACAAAG	29	18.7069	29	21.5094	1.34825	8.66532	12.7137
AAAATAAAATAAAA	72	60.5356	75	69.6577	1.07669	5.5421	12.4873
ACAAAAAAAAAAAG	21	11.6436	21	13.3863	1.56877	9.45613	12.385
AAAAGAAAAAAAAA	59	47.8313	59	55.0263	1.07221	4.11383	12.3815
TCTTTTTTTTTTTT	72	61.0267	72	70.2235	1.0253	1.79875	11.9055
CCAAAAAAAAAAAA	71	60.1056	72	69.1625	1.04103	2.89497	11.8269
CTATATATATATAG	8	1.83316	8	2.10714	3.79661	10.6729	11.7872
CACAAAAAAAAAAA	33	23.0973	33	26.5597	1.24248	7.16465	11.7741
AAAAAAAAGAAAAA	41	30.8595	41	35.4906	1.15524	5.91651	11.6492
CTCTTTTTTTTTTT	41	31.1077	41	35.7761	1.14602	5.58795	11.3208
TCAAAAAAAAAAAA	42	32.3312	42	37.1841	1.12951	5.11509	10.9887
AAATAAAATAAAAA	48	38.3463	48	44.1069	1.08827	4.0601	10.7781
AAAAAAAAAAAGAG	34	24.8901	34	28.6222	1.18789	5.85402	10.6043
AAAAAGAAAAAAAA	51	41.5409	51	47.7842	1.0673	3.32169	10.4625
AAAAAAAAAGAAAA	41	31.9395	42	36.7333	1.14338	5.62737	10.2389
AAAAAAAGAAAAAA	48	38.8326	49	44.6667	1.09701	4.53705	10.1731
TCTTTCTCTCTCTC	33	24.5148	33	28.1905	1.17061	5.19827	9.80857
AAAAAAATAAAAAA	28	19.7253	29	22.6809	1.27861	7.12747	9.80847

Table A.58: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTT	72	55.667	72	64.7945	1.11121	7.59205	18.5241
AAAAAAAAAAAAAAC	90	74.9359	90	87.2535	1.03148	2.78926	16.4859
GAAAAAAAAAAAAAA	53	42.5734	53	49.5423	1.06979	3.5757	11.6103
AAAAAAAAAAAAACT	51	41.1954	51	47.9375	1.06389	3.15831	10.8884
CAAAAACAAACAAAG	28	19.0266	28	22.1316	1.26516	6.58557	10.8182
AAAAAGAAAAAAAAA	41	31.5398	41	36.6951	1.11731	4.54806	10.7552
TTTTTTTTTTTTTTG	59	49.1999	59	57.2603	1.03038	1.76589	10.7171
TTCTCTCTCTCTCTT	13	5.90014	13	6.86134	1.89467	8.3076	10.2697
CAAAAAAAAAAAAAA	110	100.291	110	116.831	0.941531	-6.62728	10.164
ACAAAAAAAAAAAAA	51	42.1828	51	49.0874	1.03896	1.94941	9.68042
AAAAAAAATAAAAAA	16	9.1112	17	10.5962	1.60436	8.03628	9.00934
AAAAAAAAAAAAAAG	79	70.4962	79	82.0775	0.962505	-3.01904	8.9972
AAAAAAAAAAAAGAG	28	20.617	28	23.9821	1.16754	4.33706	8.57054
AAAAGAAAAAAAAAA	38	30.3278	38	35.2843	1.07697	2.81761	8.56977
AAAAAAAAAAAAAGT	30	22.5658	30	26.25	1.14286	4.00594	8.54288
ATCTCTCTCTCTCTT	10	4.32135	10	5.02521	1.98997	6.88118	8.39017
AAAAAAAAAGAAAAA	33	25.9688	33	30.2105	1.09233	2.91447	7.90714
AAAATAAAATAAAAA	36	28.9194	36	33.6449	1.07	2.43571	7.8842
TTTTTTTTTTTTTAA	25	18.269	25	21.25	1.17647	4.06297	7.8418
AATTAAAAAAAAAAG	8	3.04508	8	3.54098	2.25926	6.5203	7.7273
AAAAAAAAGAAAAAA	33	26.1655	33	30.4394	1.08412	2.66541	7.6582
TAAAAAAAAAAAAAA	47	40.0336	47	46.5845	1.00892	0.41734	7.54016
CAATCAAAAACAAAC	27	20.424	27	23.7576	1.13648	3.45426	7.53645
CACAAAAAAAAAAAA	24	17.6286	24	20.5049	1.17045	3.77741	7.4048
TTCTCTCTCTCTCTA	8	3.24951	8	3.77871	2.11712	6.00047	7.20751

Table A.59: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAAAA	87	69.3696	87	81.7143	1.06469	5.4531	19.7019
CTTTTTTTTTTTTTT	62	45.4831	62	53.5537	1.15772	9.07985	19.2072
AAAAAAAAAAAAAAAG	66	49.8377	66	58.6857	1.12463	7.75226	18.5382
AAAAAAAAAAAAAAAC	71	56.77	71	66.8571	1.06197	4.26864	15.8807
GAAAAAAAAAAAAAAA	43	33.4454	43	39.3714	1.09216	3.79087	10.8052
TTTTTTTTTTTTTTG	46	37.2764	46	43.8843	1.04821	2.1659	9.67287
AAAAAAAAAAAAAACT	40	33.0438	40	38.8983	1.02832	1.11715	7.64186
AAAAATAAAATAAAAA	17	11.0138	17	12.96	1.31173	4.61288	7.37906
CCCAAAAAAAAAAAAA	15	9.20681	15	10.8333	1.38462	4.88134	7.3216
TCTCTCTCTCTCTCTT	39	32.4476	39	38.1961	1.02105	0.812323	7.17349
CCTCTCTCTCTCTCTA	5	1.23634	5	1.45455	3.4375	6.17372	6.98642
TTTTTTTTTTTTTTTC	23	17.065	23	20.0826	1.14527	3.11968	6.8647
CCAAAAAAAAAAAAAA	37	30.755	37	36.2025	1.02203	0.806188	6.84
AAAAAAAAAAAAAAGT	24	18.1427	24	21.3514	1.12405	2.80653	6.71485
TTTTTTTCTTTTTTT	16	10.5576	16	12.4231	1.28793	4.04853	6.65186
TTTCTCTCTCTCTCTC	23	17.2341	23	20.2817	1.13403	2.89284	6.63793
AAGAGAGAGAGAGAGC	4	0.788941	4	0.928177	4.30952	5.84331	6.49343
TAAAAAAAAAAAAAT	6	2.04499	6	2.40595	2.49381	5.48288	6.4582
TTAAAAAAAAAAAAA	24	18.3836	24	21.6349	1.10932	2.48988	6.39831
GATATATATATATATT	3	0.393364	3	0.462783	6.48252	5.60733	6.0949
CAAAGAAGAAGAAGAG	4	0.894724	4	1.05263	3.8	5.34	5.99014
GAAAGAAAAAAAAAAC	6	2.22942	6	2.62295	2.2875	4.96476	5.9401
AAAAAAAAAAAAAACA	26	20.738	26	24.4068	1.06528	1.64413	5.87936
CTTCTTCTTCTTCCTT	11	6.44754	11	7.58621	1.45	4.0872	5.87617
TTAAAATAAAATAAAA	12	7.3657	12	8.66667	1.38462	3.90507	5.85687

Table A.60: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAAAA	74	53.236	74	63.4375	1.1665	11.3967	24.3704
AAAAAAAAAAAAAAAC	56	43.4532	56	51.7708	1.08169	4.39739	14.2053
TTTTTTTTTTTTTTTG	39	28.3215	39	33.7333	1.15613	5.65793	12.4777
CTTTTTTTTTTTTTTT	48	38.1656	48	45.4667	1.05572	2.60263	11.0048
AAAAAAAAAAAAAAAG	50	40.3954	50	48.125	1.03896	1.91106	10.6654
TAAAAAAAAAAAAAAA	28	20.8172	28	24.7917	1.12941	3.40751	8.29994
ACAAAAAAAAAAAAAA	28	21.2515	28	25.3091	1.10632	2.82914	7.72179
AAAAAAAAAAAAAAAT	25	18.3689	25	21.875	1.14286	3.33828	7.70541
CAGAGAGAGAGAGAGAT	4	0.797459	4	0.949367	4.21333	5.75302	6.45048
TTTCTTCTTCTTCTTCT	20	14.7452	20	17.5584	1.13905	2.60395	6.09633
GTTTTTTTTTTTTTTT	27	21.5516	27	25.6667	1.05195	1.36738	6.08544
AAAAAAAAAAAAAAAGA	30	24.5511	30	29.2405	1.02597	0.769273	6.0132
AAGAGAGAGAGAGAGAA	4	0.935683	4	1.11392	3.59091	5.11362	5.81109
CGTTTTTTTTTTTTTC	3	0.444703	3	0.529412	5.66667	5.2038	5.72688
AATCAAAAACAAACAAA	26	21.0231	26	25.037	1.03846	0.981249	5.52431
CAATCAAAAACAAACAA	26	21.052	26	25.0714	1.03704	0.945559	5.48864
AAAAAAAAGAAAAAAAA	18	13.2729	18	15.8049	1.13889	2.34096	5.48362
AACCAAAAAAAAAAAAC	7	3.26381	7	3.88571	1.80147	4.12022	5.34109
GAAAAAAAAAAAAAAC	8	4.10951	8	4.89263	1.63511	3.9337	5.3291
CCACACACACACACACC	2	0.152727	2	0.181818	11	4.79579	5.1445
AGACTTTTTACTCAATT	29	24.3493	29	29	1	3.93441e-12	5.06902
GACTTTTTACTCAATTT	29	24.3493	29	29	1	3.93441e-12	5.06902
CTCTCTTCTTCTTCTTC	13	8.81861	13	10.5	1.2381	2.77646	5.04511
AGAAAAAAAAAAAAAA	25	20.4377	25	24.3396	1.02713	0.669256	5.03732
ATCTTTTTTTTTTTTA	3	0.559996	3	0.666667	4.5	4.51223	5.03532

Table A.61: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAAAAAA	51	43.567	51	52.5319	0.970839	-1.50931	8.03379
AAAAAAAAAAAAAAAAAC	40	32.976	40	39.7538	1.00619	0.246914	7.72399
CTTTTTTTTTTTTTTTTT	36	29.08	36	35.0545	1.02697	0.958091	7.6849
TTCTCTCTCTCTCTCTCA	5	1.0849	5	1.30713	3.82519	6.70804	7.63977
ACAAAAAAAAAAAAAAAA	26	19.7603	26	23.8161	1.0917	2.28111	7.13496
CAAAAAAAAAAAAAAAAG	12	6.74864	12	8.13187	1.47568	4.66939	6.90679
TTTTTTTTTTTTTTTTTG	29	23.6298	29	28.4818	1.01819	0.522867	5.93881
AAAAAAAAAAAAAAAAGA	24	18.8572	24	22.7273	1.056	1.30772	5.78779
GTCTCTCTCTCTCTCTCC	3	0.440489	3	0.530713	5.65278	5.19644	5.75545
AGACTTTTTACTCAATTT	29	24.0595	29	29	1	3.92153e-12	5.41617
CTCTTCTTCTTCTTCTTC	14	9.54337	14	11.5	1.21739	2.75394	5.36495
GTTTTTTTTTTTTTTTTT	21	16.3613	21	19.7182	1.06501	1.32261	5.24172
AAAAAAAAAGAAAAAAAA	16	11.6176	16	14	1.14286	2.1365	5.12111
TTTACTCAATTTCTCATT	27	22.4009	27	27	1	3.65707e-12	5.04183
GACTTTTTACTCAATTTC	27	22.4009	27	27	1	3.65707e-12	5.04183
CTTTTTACTCAATTTCTC	27	22.4009	27	27	1	3.65707e-12	5.04183
AAAAACAAACAAAGCATT	27	22.4009	27	27	1	3.65707e-12	5.04183
AAACAAAGCATTAAGCAT	27	22.4009	27	27	1	3.65707e-12	5.04183
AAAACAAACAAAGCATTA	27	22.4009	27	27	1	3.65707e-12	5.04183
ACTTTTTACTCAATTTCT	27	22.4009	27	27	1	3.65707e-12	5.04183
AAACAAACAAAGCATTAA	27	22.4009	27	27	1	3.65707e-12	5.04183
AACAAAGCATTAAGCATA	27	22.4009	27	27	1	3.65707e-12	5.04183
ACAAAGCATTAAGCATAT	27	22.4009	27	27	1	3.65707e-12	5.04183
ATCAATCAAAAACAAACA	27	22.4009	27	27	1	3.65707e-12	5.04183
CAAAAACAAACAAAGCAT	27	22.4009	27	27	1	3.65707e-12	5.04183

Table A.62: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTCTCTCTCTCTCTCTCTT	6	1.10871	6	1.35211	4.4375	8.94055	10.1314
CTTTTTTTTTTTTTTTTTT	29	21.305	29	25.9917	1.11574	3.17604	8.94227
TTTTTTTTTTTTTTTTTTG	24	17.1637	24	20.9378	1.14625	3.27599	8.04624
CAAAAAAAAAAAAAAAAAA	37	30.1509	37	36.7895	1.00572	0.211128	7.574
TCAAAAAAAAAAAAAAAAA	12	7.3458	12	8.95946	1.33937	3.50636	5.88934
ATATATATATATATATATT	18	13.1133	18	15.9955	1.12531	2.12511	5.70147
AGCAAGGATTTTATTGTTA	23	18.0683	23	22.0417	1.04348	0.978871	5.55074
TTTTACTCAATTTCTCATT	27	22.1312	27	27	1	3.65707e-12	5.36897
TCAAAAACAAACAAAGCAT	27	22.1312	27	27	1	3.65707e-12	5.36897
AAACAAAGCATTAAGCATA	27	22.1312	27	27	1	3.65707e-12	5.36897
CTTTTTACTCAATTTCTCA	27	22.1312	27	27	1	3.65707e-12	5.36897
AAAACAAACAAAGCATTAA	27	22.1312	27	27	1	3.65707e-12	5.36897
AACAAAGCATTAAGCATAT	27	22.1312	27	27	1	3.65707e-12	5.36897
AGACTTTTTACTCAATTTC	27	22.1312	27	27	1	3.65707e-12	5.36897
ACTTTTTACTCAATTTCTC	27	22.1312	27	27	1	3.65707e-12	5.36897
AAAAACAAACAAAGCATTA	27	22.1312	27	27	1	3.65707e-12	5.36897
CAAAAACAAACAAAGCATT	27	22.1312	27	27	1	3.65707e-12	5.36897
GACTTTTTACTCAATTTCT	27	22.1312	27	27	1	3.65707e-12	5.36897
TTTTTACTCAATTTCTCAT	27	22.1312	27	27	1	3.65707e-12	5.36897
CAAAAAAAAAAAAAAAAAT	6	2.4599	6	3	2	4.15888	5.34984
GTTTTTTTTTTTTTTTTTT	17	12.4299	17	15.1618	1.12124	1.94535	5.32281
CAATCAAAAACAAACAAAG	26	21.3118	26	26	1	3.52163e-12	5.16973
ACTAAGACTTTTTACTCAA	26	21.3118	26	26	1	3.52163e-12	5.16973
AAGACTTTTTACTCAATTT	26	21.3118	26	26	1	3.5274e-12	5.16973
AAACAAACAAAGCATTAAG	26	21.3118	26	26	1	3.51585e-12	5.16973

Table A.63: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAAAAAAAAAAAAAAAC	24	16.64	24	20.5494	1.16792	3.72538	8.78988
CAAAAAAAAAAAAAAAAAAA	29	21.9864	29	27.1545	1.06796	1.90683	8.02925
TTCTCTCTCTCTCTCTCTCC	3	0.28495	3	0.351792	8.52778	6.42999	7.06216
ATTTTATTGTTATTCATCAA	24	17.9388	24	22.1538	1.08333	1.92102	6.98609
CCAAAAAAAAAAAAAAAAAA	12	7.05088	12	8.70588	1.37838	3.85089	6.38105
CTTTTTTTTTTTTTTTTTTT	22	16.4651	22	20.3333	1.08197	1.73318	6.37557
CTATATATATATATATATAC	3	0.394873	3	0.4875	6.15385	5.45123	6.08341
AAAAAAAAAAAAAAAAAACT	14	9.07053	14	11.2	1.25	3.12401	6.07637
GACTTTTTACTCAATTTCTC	27	21.8614	27	27	1	3.65707e-12	5.70012
CTTTTTACTCAATTTCTCAT	27	21.8614	27	27	1	3.65707e-12	5.70012
CAAAAACAAACAAAGCATTA	27	21.8614	27	27	1	3.65707e-12	5.70012
ACTTTTTACTCAATTTCTCA	27	21.8614	27	27	1	3.65707e-12	5.70012
AAACAAAGCATTAAGCATAT	27	21.8614	27	27	1	3.65707e-12	5.70012
AAAAACAAACAAAGCATTAA	27	21.8614	27	27	1	3.65707e-12	5.70012
AGACTTTTTACTCAATTTCT	27	21.8614	27	27	1	3.65707e-12	5.70012
TCAAAAACAAACAAAGCATT	27	21.8614	27	27	1	3.65707e-12	5.70012
TTTTTACTCAATTTCTCATT	27	21.8614	27	27	1	3.65707e-12	5.70012
TTTTTATTGTTATTCATCAC	2	0.124615	2	0.153846	13	5.1299	5.55134
CTAAGACTTTTTACTCAATT	26	21.052	26	26	1	3.53317e-12	5.48862
ATCAATCAAAAACAAACAAA	26	21.052	26	26	1	3.53317e-12	5.48862
AAACAAACAAAGCATTAAGC	26	21.052	26	26	1	3.53317e-12	5.48862
ACAAACAAAGCATTAAGCAT	26	21.052	26	26	1	3.53317e-12	5.48862
AAAACAAACAAAGCATTAAG	26	21.052	26	26	1	3.52163e-12	5.48862
AACAAACAAAGCATTAAGCA	26	21.052	26	26	1	3.53317e-12	5.48862
ACTAAGACTTTTTACTCAAT	26	21.052	26	26	1	3.53317e-12	5.48862

A.4 The *Arabidopsis thaliana* Proximal Promoters

A.4.1 Timing Information

Table A.64: Full Size and Timing Information, Proximal Promoters

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	2.788834	4.027980	42.700026	6.816814
2	0	0.000001	5.148080	2.994285	76.191952	8.142365
3	1	0.000003	8.229275	3.116404	55.660069	11.345679
4	2	0.000012	12.254478	2.550311	16.463949	14.804789
5	3	0.000046	16.434215	3.471859	3.578265	19.906074
6	4	0.000186	21.011225	6.202330	2.462631	27.213555
7	5	0.000743	27.252092	5.872936	1.630637	33.125028
8	6	0.002971	34.704883	17.074537	1.756133	51.77942
9	7	0.011875	44.938118	71.141537	2.875015	116.079655
10	8	0.046703	57.245912	217.210981	8.470263	274.456893
11	9	0.167589	70.828641	498.336183	27.410219	569.164824
12	10	0.494220	87.310197	1239.760385	69.595830	1327.070582
13	11	1.146875	103.558001	2105.993635	130.030991	2209.551636
14	12	2.141166	122.320368	3282.601786	189.291383	3404.922154
15	13	3.384761	137.566645	2862.979541	239.218588	3000.546186
16	14	4.766900	154.110458	4224.087650	279.880857	4378.198108
17	15	6.213774	169.427025	3463.833468	315.198931	3633.260493
18	16	7.688418	187.645451	3733.323573	346.986109	3920.969024
19	17	9.174630	205.004500	4239.962195	380.974356	4444.966695
20	18	10.665619	222.807821	3022.154789	413.819507	3244.96261

A.4.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.65: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
G	27169	27169	3997934	3.99793e+06	1	8.8772e-10	0
C	27169	27169	4014441	4.01444e+06	1	8.91385e-10	0
T	27169	27169	8137523	8.13752e+06	1	0	0
A	27169	27169	8301377	8.30138e+06	1	0	0

Table A.66: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CG	27165	27169	529803	656386	0.807152	-113507	-3.99919
AC	27169	27169	1270207	1.36293e+06	0.931968	-89495.3	1.18953e-07
AG	27169	27169	1320673	1.35733e+06	0.972996	-36153.8	1.27254e-07
CC	27169	27169	708189	659096	1.07449	50877.7	0.000502811
CA	27169	27169	1472689	1.36293e+06	1.08053	114064	1.18953e-07
GC	27168	27169	598021	656386	0.911082	-55689.4	-0.999463
GG	27168	27169	702686	653687	1.07496	50791.4	-0.999446
GA	27169	27169	1456486	1.35733e+06	1.07306	102697	1.27254e-07
AT	27169	27169	2482407	2.76275e+06	0.898529	-265608	0
CT	27169	27169	1299042	1.33603e+06	0.972316	-36469.9	1.64428e-07
AA	27169	27169	3218655	2.81837e+06	1.14203	427448	0
TA	27169	27169	2144613	2.76275e+06	0.776262	-543157	0
GT	27169	27169	1236515	1.33054e+06	0.929337	-90617	1.75655e-07
TC	27169	27169	1433461	1.33603e+06	1.07293	100902	1.64428e-07
TG	27169	27169	1439941	1.33054e+06	1.08223	113786	1.75655e-07
TT	27169	27169	3110710	2.70821e+06	1.14862	431025	0

Table A.67: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ACT	27169	27169	374646	411029	0.911483	-34723.1	0.0157779
GAT	27169	27169	422228	435541	0.969433	-13107.5	0.00954299
ATC	27169	27169	428970	437287	0.98098	-8237.47	0.00923768
CAT	27169	27169	445250	440386	1.01104	4890.36	0.00872813
TCT	27169	27169	518733	463857	1.1183	58001.2	0.00588674
ACA	27169	27169	489625	465973	1.05076	24242.7	0.0056967
GTT	27169	27169	476500	472679	1.00808	3835.99	0.00514642
AAC	27169	27169	499014	492492	1.01324	6565.44	0.00388172
AAG	27169	27169	509020	512059	0.994066	-3029.48	0.00299544
TGA	27169	27169	480881	524584	0.916689	-41830.2	0.00255484
TTC	27169	27169	519393	547965	0.947857	-27814.2	0.00191589
TTG	27169	27169	567759	550443	1.03146	17586	0.00185926
GAA	27169	27169	537388	564717	0.951606	-26656.5	0.0015663
TAT	27169	27169	736698	641316	1.14873	102148	0.00063597
ATA	27169	27169	747800	654229	1.14303	99964.7	0.000546934
TTA	27169	27169	746702	819816	0.910817	-69752	7.89517e-05
TAA	27169	27169	762739	831521	0.917282	-65855.4	6.8827e-05
AAT	27169	27169	875568	962492	0.909688	-82875.4	1.47517e-05
AAA	27169	27169	1331042	1.24795e+06	1.06658	85794.1	4.98732e-07
TAG	27168	27168.9	326150	341188	0.955924	-14701.9	-0.889663
AGT	27168	27169	371250	408469	0.908882	-35469.3	-0.983264
TGT	27168	27169	473638	445357	1.0635	29160.3	-0.991993
AGA	27168	27169	537706	481134	1.11758	59774.8	-0.995432
TCA	27168	27169	487156	525862	0.926395	-37245.3	-0.997467
TTT	27168	27169	1273459	1.18912e+06	1.07092	87258.3	-0.999981

Table A.68: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTGC	17604	16918.5	34184	26557.4	1.28717	8629.67	699.196
GTCG	17286	16750.2	31954	26114	1.22363	6449.1	544.248
CGAC	17538	17004.5	32320	26786.9	1.20656	6068.82	541.759
GCAG	17670	17180.2	36734	27261.8	1.34745	10954.7	496.668
GTAG	22895	22483.6	54333	47866.9	1.13508	6884.37	415.139
CGCG	8089	7735.32	11291	9132.26	1.23639	2395.87	361.647
GACG	17627	17386.8	34692	27830.7	1.24654	7645.11	241.892
CGTC	17397	17192.8	33034	27296.1	1.21021	6302.7	205.392
CTAC	22881	22706	55120	49189.9	1.12056	6274.04	175.699
CGCC	10971	10857	17566	13903.5	1.26342	4107.36	114.557
GGCG	11044	10958.9	17667	14074.2	1.25528	4016.73	85.4296
TGCA	25034	24970.1	76867	68441.8	1.1231	8923.74	63.9702
TAGT	26278	26234.1	108976	91682.9	1.18862	18830.1	43.9729
ACTA	26369	26373.9	112254	96082.2	1.16831	17462.2	-4.86177
AAAT	27160	27168.9	358703	362082	0.990667	-3363.41	-8.94034
CAAA	27144	27165.3	251375	241600	1.04046	9969.82	-21.248
ATTT	27146	27168.9	349404	352853	0.990225	-3432.17	-22.9131
GATT	27021	27045.4	153000	146603	1.04364	6534.7	-24.3892
AAAA	27142	27169	557506	550439	1.01284	7112.31	-26.9847
TGAT	26980	27007.8	156557	139405	1.12304	18166.4	-27.8259
GAAA	27133	27161.4	217005	222231	0.976482	-5164.39	-28.3662
TTTG	27135	27163.8	240792	232429	1.03598	8512.15	-28.7454
AATT	27139	27168.6	302786	304008	0.99598	-1219.79	-29.5889
AAAC	27124	27155.3	216907	206362	1.0511	10809.8	-31.3228
TTTC	27124	27158.2	210217	212629	0.988658	-2397.8	-34.1328

Table A.69: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GATTC	17157	15663.5	29031	23438.3	1.23862	6212.49	1562.52
GAATC	17222	15872	29557	23936.7	1.2348	6233.75	1405.89
CGCCG	3745	2801.46	5135	2969.68	1.72914	2812.06	1087.1
CGGCG	3761	2818.86	5244	2989.18	1.75433	2947.58	1084.49
CGTCG	5622	4667.07	7368	5143.02	1.43262	2648.84	1046.57
CAAGT	17120	16144.6	28093	24602.7	1.14187	3726.92	1004.31
TGAGT	14975	14012.1	22808	19781.2	1.15301	3247.34	995.215
AGGTA	11228	10297.7	14984	12999.4	1.15267	2128.95	971.134
CGACG	5724	4866.9	7553	5386.41	1.40223	2553.41	928.491
GTTAC	13459	12614.5	19216	17028.1	1.12849	2322.82	872.187
GTAAC	13584	12752.8	19359	17288.5	1.11976	2189.78	857.709
GGGCC	4713	3965.26	5882	4305.03	1.36631	1835.85	814.186
TAGGG	7622	6857.91	9382	7937.75	1.18195	1568.33	805.153
TTAGG	13892	13120	20425	17992.1	1.13522	2590.41	794.29
TAGAT	19330	18552.4	35995	31320.3	1.14926	5007.45	793.639
ACTCG	8147	7420.02	10107	8703.39	1.16127	1511.15	761.483
GGACC	6341	5625.6	7548	6330.74	1.19228	1327.43	759.069
TGAAT	22106	21376.4	48191	42141.3	1.14356	6464.48	741.954
TACCT	11146	10433.1	14718	13219.3	1.11338	1580.66	736.678
GGCCC	4537	3881.17	5617	4206.32	1.33537	1624.49	708.361
ACTCA	15376	14696.3	23784	21237.3	1.11992	2693.66	695.231
TGGGC	7926	7265	11103	8490.09	1.30776	2979.11	690.197
ACTTG	16958	16293	28034	24972.2	1.12261	3242.22	678.395
GATAC	12009	11350.3	16300	14756.5	1.1046	1621.58	677.475
GTACG	5744	5124.68	6756	5703.61	1.18451	1144.01	655.32

Table A.70: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAC	13854	12339.6	20348	16536.2	1.23051	4220.79	1603.79
GTTTTG	13059	11641.8	18878	15280.8	1.23541	3990.81	1500.18
TGAAAT	13301	12352.3	18824	16559.7	1.13674	2412.53	984.241
TAGAAT	8488	7592.08	10602	8952.1	1.1843	1793.38	946.822
GATAAG	5565	4702.47	6423	5191.78	1.23715	1366.87	937.192
TGAAGT	6952	6078.94	8194	6918.7	1.18433	1386.21	932.954
GATTTG	10089	9199.99	13216	11292.6	1.17032	2078.61	930.646
ATTCTA	8606	7744.2	10666	9165.14	1.16376	1617.54	908.067
CTAAAC	7818	6967.79	9711	8094.76	1.19966	1767.81	900.09
CAAATC	10618	9760.18	14034	12157.5	1.15435	2014.39	894.457
ACTTCA	7051	6215.6	8351	7096.27	1.17682	1359.64	889.177
CTTATC	5706	4907.01	6622	5441.59	1.21692	1300.07	860.774
GTTTAG	7251	6455.8	8834	7411.18	1.19198	1551.41	842.275
GATATG	6307	5534.69	7407	6222.79	1.1903	1290.35	823.849
GATTAG	6048	5291.3	7078	5917.22	1.19617	1267.85	808.398
CATATC	6429	5680.57	7533	6407.59	1.17564	1218.92	795.705
GAATTG	6896	6147.48	8172	7007.61	1.16616	1256.18	792.348
GCTAAG	3045	2353.52	3255	2475.39	1.31494	891.202	784.353
CTAATC	6355	5623.96	7479	6335.73	1.18045	1240.72	776.62
GAAATG	8368	7631.4	10325	9007.01	1.14633	1410.03	771.057
ATTTCA	13406	12674.3	19187	17159.5	1.11815	2142.79	752.445
CAATTC	7132	6444.45	8502	7396.21	1.14951	1184.62	722.989
GCTTTG	5684	5008.06	6640	5565.86	1.19299	1171.7	719.633
CACGTG	3584	2944.67	4331	3134.01	1.38193	1401.01	704.198
CATTTC	8567	7892.77	10525	9374.83	1.12269	1218.01	702.245

Table A.71: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTG	5740	4379.01	6608	4806.68	1.37475	2103.16	1553.45
CAAAAAC	6136	4770.09	7314	5280	1.38523	2383.37	1545.1
GAAAAAG	5538	4311.25	6384	4725.49	1.35097	1920.45	1386.74
CTTTTTC	5305	4179.61	6069	4568.45	1.32846	1723.72	1264.88
ATTTTTA	9675	8593.03	13306	10397	1.27979	3282.54	1147.4
TAAAAAT	10066	9026.09	14102	11041.9	1.27713	3449.57	1097.64
GAAAATG	4575	3744.59	5135	4055.82	1.26608	1211.49	916.349
GAAGATG	3184	2434.09	3568	2567.09	1.3899	1174.7	855.117
GATTTTG	5092	4330.72	5832	4748.79	1.2281	1198.3	824.583
CATTTTC	4614	3870.26	5166	4202.93	1.22914	1065.83	811.017
GTTCTTG	2279	1616.03	2470	1677.21	1.47269	956.106	783.443
CAAAATC	5391	4695.68	6295	5189.32	1.21307	1215.9	744.429
GAACAAG	2252	1633.04	2388	1695.42	1.4085	817.953	723.743
GAGAAAG	3701	3044.08	4237	3249.99	1.3037	1123.67	723.187
CAACAAC	3210	2564.87	3994	2712.07	1.47267	1546	720.207
GATGAAG	2618	2010.34	2914	2102.53	1.38595	951.081	691.427
CAAAACC	4088	3468.4	4573	3735.26	1.22428	925.358	671.919
CATCTTC	2877	2284.55	3201	2402.25	1.3325	918.872	663.375
GAAACAG	2437	1859.3	2659	1938.83	1.37145	839.889	659.377
GAAATTG	3532	2937.03	3869	3128.9	1.23654	821.445	651.532
CATAAAC	3022	2441.43	3264	2575.2	1.26747	773.648	644.703
CTTTCTC	3456	2870.78	3863	3054.23	1.2648	907.482	641.187
CAAGAAC	2181	1628.98	2341	1691.07	1.38433	761.326	636.481
GTTTATG	2858	2290.48	3064	2408.77	1.27202	737.217	632.647
TCAAAAT	7504	6897.77	9107	8009.21	1.13707	1169.81	632.122

Table A.72: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAAAAAAT	4282	3436.7	4873	3702.85	1.31601	1338.15	941.645
ATTTTTTA	3911	3167.35	4419	3393.88	1.30205	1166.35	824.818
TTATATAA	3107	2521.71	3405	2667.13	1.27665	831.645	648.491
AATATATT	3652	3127.75	4112	3348.74	1.22793	844.301	565.921
GAAAAAAG	2076	1660.65	2192	1726.94	1.2693	522.712	463.436
CTTTTTTC	1958	1581.48	2072	1642.09	1.26181	481.832	418.16
AAAAATTG	3001	2614.58	3239	2770.48	1.16911	506.069	413.662
TAAAATTT	4375	3985.7	5110	4343.62	1.17644	830.335	407.723
TAATTTTT	4682	4297.55	5364	4714.38	1.1378	692.451	401.153
CAATTTTT	2878	2511.2	3117	2655.46	1.17381	499.511	392.367
AAATTTTA	4276	3910.03	4965	4254.4	1.16703	766.889	382.588
TACAAAAT	2605	2260.52	2839	2378.51	1.1936	502.437	369.487
ATTTTCTA	2213	1901.21	2356	1986.38	1.18608	402.057	336.062
TGAAAAAT	2376	2082.21	2518	2183.21	1.15335	359.243	313.609
CATTTTTC	1696	1436.35	1782	1487.23	1.1982	322.223	281.825
AAAAAATC	3901	3635.01	4293	3932.58	1.09165	376.454	275.498
TAAGAAAT	1901	1653.73	2000	1719.52	1.16312	302.205	264.894
TAGAAAAT	2170	1921.47	2299	2008.34	1.14473	310.749	263.954
ATTCTTCA	1172	935.852	1196	959.796	1.2461	263.14	263.712
CAAAATTT	3360	3107.4	3672	3325.58	1.10417	363.872	262.599
GGAAAAAA	2689	2439.93	2865	2576.43	1.112	304.156	261.37
AAAAATTA	4773	4519.15	5593	4980.92	1.12288	648.231	260.848
ATTTTTCA	2343	2099.16	2494	2201.72	1.13275	310.872	257.478
ATTTTGTA	2472	2228.13	2690	2342.93	1.14814	371.596	256.758
TTTTTTGG	3380	3132.76	3738	3354.45	1.11434	404.69	256.749

Table A.73: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTA	2140	1487.7	2295	1543.65	1.48673	910.154	778.054
TAAAAAAAT	2219	1624.19	2390	1689.73	1.41443	828.671	692.428
AATATATAA	1546	1069.53	1631	1100.92	1.48149	641.064	569.62
TTATATATT	1449	1043.09	1521	1073.16	1.41732	530.471	476.269
ATTTTCTTA	1204	921.892	1245	946.289	1.31567	341.557	321.44
TAAGAAAAT	1197	940.554	1239	965.786	1.28289	308.657	288.602
AAAAATTTA	1462	1213.62	1523	1252.67	1.2158	297.603	272.217
GAAAAAAAT	1533	1291.26	1604	1334.79	1.20169	294.703	263.071
TTATATATC	605	395.229	611	401.7	1.52104	256.249	257.586
AAAATTTTA	1682	1455.09	1777	1508.88	1.1777	290.648	243.743
TAAATTTTT	1418	1194.85	1497	1232.85	1.21426	290.618	242.797
TGAAAAAAT	871	665.177	892	679.486	1.31276	242.74	234.812
TTCTCTCTT	837	633.203	870	646.437	1.34584	258.406	233.55
GATTTTTTT	2115	1897.06	2227	1984.1	1.12242	257.194	230.006
TAAAATTTT	1680	1470.36	1791	1525.16	1.1743	287.77	223.916
AAGAGAGAA	891	697.481	926	712.917	1.29889	242.157	218.178
ATTTTTTTC	1429	1231.08	1489	1271.11	1.17141	235.578	213.037
TTTTTCTTT	3671	3465.13	4169	3739.86	1.11475	452.872	211.866
GAGAAAGAG	683	502.049	724	511.285	1.41604	251.853	210.225
GAAAAAAAG	987	803.071	1016	822.477	1.23529	214.69	203.546
AGTTTTTTT	2229	2035.99	2351	2135.2	1.10107	226.353	201.885
CATATATAC	488	323.436	501	328.293	1.52608	211.773	200.721
ATTTTTCTA	830	655.05	857	669.014	1.28099	212.222	196.473
AAAAAAATC	2312	2125.6	2432	2233.11	1.08906	207.494	194.342
GATATATAA	568	406.654	584	413.399	1.41268	201.764	189.802

Table A.74: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTA	1036	690.564	1065	706.547	1.50733	437.012	420.216
TAAAAAAAAT	1078	748.72	1117	766.887	1.45654	420.062	392.929
AATATATATT	739	445.706	756	453.938	1.66543	385.621	373.666
TTATATATAA	541	320.103	558	325.254	1.71558	301.182	283.904
CATATATATG	313	150.473	316	152.415	2.07328	230.406	229.247
TTTTTAAAAA	1756	1544.4	1863	1606.05	1.15999	276.494	225.472
GATTTTTTTT	1312	1138.32	1360	1174.57	1.15787	199.349	186.302
AAAAAAAATC	1387	1221.78	1433	1262.7	1.13487	181.297	175.916
AAAGAAGAAA	751	599.836	767	612.676	1.25189	172.307	168.787
TTTAAAAAAT	886	740.821	923	758.684	1.21658	180.948	158.556
ATTAAAAAAA	1475	1329.46	1531	1376.82	1.11198	162.509	153.232
GTATATATAG	203	95.5322	205	96.6671	2.12068	154.106	153.01
AAAAAAAATT	1835	1695.08	1929	1767.89	1.09113	168.241	145.537
AATATATATC	260	151.279	260	153.234	1.69675	137.466	140.804
ATTTTTTTGA	365	250.351	368	254.051	1.44853	136.362	137.617
ATTTTTTAAA	809	683.087	836	698.799	1.19634	149.866	136.864
AAAAAAAACT	1505	1375.81	1561	1426.09	1.09461	141.105	135.077
CTATATATAA	271	164.832	271	167.003	1.62272	131.193	134.74
TTATATATAC	286	179.496	291	181.91	1.5997	136.716	133.23
AGTTTTTTTT	1517	1392.61	1574	1443.96	1.09005	135.722	129.79
AAATTTTTTA	419	307.493	426	312.368	1.36378	132.17	129.647
TAAAAAATTT	421	311.775	430	316.743	1.35757	131.449	126.448
ATTTTAAAAT	906	788.94	947	808.696	1.17102	149.509	125.344
ATATTTTATA	588	479.07	607	488.222	1.24329	132.18	120.469
AATATATATG	355	253.235	359	256.991	1.39693	120.006	119.919

Table A.75: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAAAAAAAAAT	607	394.792	620	402.154	1.5417	268.388	261.113
ATTTTTTTTTA	558	362.181	575	368.711	1.55949	255.506	241.176
AATATATATAA	289	134.32	293	136.165	2.1518	224.527	221.433
TTATATATATT	290	149.956	296	152.06	1.9466	197.161	191.267
TTAAAAAAAAA	1368	1225.83	1421	1268.41	1.1203	161.426	150.114
AAAAAAAAATC	923	801.767	947	822.966	1.15072	132.944	129.969
TTTTTTTTTAA	1295	1177.13	1342	1216.89	1.10281	131.333	123.578
TTATATATATG	153	78.9461	154	79.9492	1.92622	100.956	101.236
GATTTTTTTTT	858	765.239	879	784.931	1.11984	99.4929	98.1694
CATATATATAA	147	78.8268	150	79.8282	1.87904	94.6138	91.6074
GTATATATATT	159	91.5928	162	92.7782	1.7461	90.2963	87.6967
GTATATATATG	108	48.1955	109	48.7803	2.23451	87.6383	87.1414
CTATATATATT	138	74.2713	139	75.2085	1.8482	85.3752	85.4951
ATTTTTTTCTA	174	106.491	174	107.899	1.61262	83.1478	85.4331
TTTTTTAAAAA	860	780.231	881	800.536	1.10051	84.3792	83.7143
TAAAAGAAAAT	204	137.606	204	139.506	1.46231	77.5233	80.3197
CTTTTATTTTC	129	70.6224	129	71.5088	1.80397	76.109	77.718
AATATATATAC	135	76.3969	135	77.364	1.745	75.1616	76.8599
CATATATATAC	98	44.8145	98	45.3554	2.16071	75.503	76.6787
TTTTTTTTATC	308	242.055	311	245.871	1.26489	73.081	74.2075
ATAAAAAAAAT	454	387.888	467	395.07	1.18207	78.1132	71.4507
AAAGAAGAAAA	402	337.183	410	343.103	1.19498	73.0319	70.6818
TATATAAAAAA	428	363.38	437	369.939	1.18127	72.8016	70.0528
TATTTTTTTTA	379	315.065	383	320.465	1.19514	68.2748	70.0231
GTTTTATTTTG	145	90.4902	147	91.6595	1.60376	69.4357	68.3663

Table A.76: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTC	293	159.282	296	161.727	1.83025	178.917	178.582
GAAAAAAAAAAC	278	158.026	282	160.448	1.75758	159.03	157.031
TAAAAAAAAAAT	328	205.862	329	209.202	1.57265	148.958	152.785
ATTTTTTTTTTA	290	190.844	292	193.887	1.50604	119.568	121.343
TTATATATATAA	122	48.1518	124	48.7908	2.54146	115.66	113.419
ATTTTTTTTTTC	261	169.089	263	171.715	1.5316	112.121	113.299
AATATATATATT	145	72.4077	146	73.4014	1.98906	100.399	100.691
TTAAAAAAAAAA	922	830.757	941	854.148	1.10168	91.1251	96.0799
GAAAAAAAAAAG	232	154.302	235	156.656	1.5001	95.3001	94.6154
AAAAAAATAAAA	591	506.438	608	517.538	1.17479	97.9444	91.2589
CTTTTTTTTTTC	227	151.861	229	154.17	1.48537	90.6069	91.2511
TTTTTTTTTTAA	873	793.771	899	815.552	1.10232	87.5789	83.0579
CTATATATATAA	74	24.563	74	24.8781	2.9745	80.6657	81.6091
GAAAAAAAAAAT	234	168.121	235	170.729	1.37645	75.0843	77.3696
TAAAAAATAAAT	171	113.815	173	115.465	1.4983	69.9487	69.6129
CTTCTTCTTTTT	254	194.931	257	198.053	1.29763	66.959	67.2315
AAAAAAAAAATC	589	528.941	597	540.762	1.104	59.0664	63.3461
TTATATATATAC	69	28.2398	69	28.6041	2.41224	60.7584	61.6428
GTTAAAATCTAG	82	41.0003	84	41.5389	2.0222	59.1516	56.8374
AAAAAGAAGAAG	266	215.04	272	218.566	1.24447	59.49	56.5705
TTTTTTTTTTGA	391	340.726	396	347.12	1.14082	52.1702	53.8135
TTATATATATAG	55	20.7415	55	21.0061	2.61828	52.9386	53.6359
AAAAATAATAAT	284	235.366	287	239.315	1.19926	52.1481	53.3445
TCAAAAAAAAAA	421	372.007	429	379.209	1.1313	52.9256	52.0865
AATATATATATC	54	20.8285	54	21.0943	2.55993	50.7589	51.4436

Table A.77: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTTTTA	162	98.082	163	99.5871	1.63676	80.3129	81.2903
TAAAAAAAAAAAT	159	102.488	159	104.069	1.52783	67.3923	69.8265
TTAAAAAAAAAAA	604	538.505	611	551.257	1.10838	62.8688	69.3255
TAATAAAATAAAT	81	37.6322	81	38.1672	2.12224	60.9503	62.0937
AAAAAATAAAAAA	297	243.032	310	247.423	1.25292	69.8967	59.5605
GAAAAAAAAAAAC	115	68.8804	117	69.8997	1.67383	60.2681	58.9445
TTATATATATATT	71	31.6249	71	32.0709	2.21384	56.4258	57.4201
ACAAAAAAAAAAC	166	118.136	167	119.993	1.39174	55.2031	56.465
TCAAAAAAAAAAA	328	278.397	333	283.613	1.17414	53.4575	53.7811
TTTCTTCTTCTTT	114	71.8759	119	72.9437	1.6314	58.2429	52.5833
TTTTTTTTTTTGA	308	262.023	311	266.851	1.16544	47.6149	49.7939
AAAAAAATAAAAA	367	320.941	376	327.213	1.1491	52.2562	49.2162
AATGTATATATAG	53	24.3247	55	24.6644	2.22993	44.1084	41.2764
GTATATATATATT	48	20.4174	48	20.701	2.31872	40.3688	41.0311
GAAAAAAAAAAAG	106	72.3445	109	73.4198	1.48461	43.0717	40.4919
AAAAAAAAAAATC	376	337.761	379	344.468	1.10025	36.2073	40.3264
ATTTTTTTTAAAA	113	79.6049	113	80.799	1.39853	37.9029	39.5852
TTTTTATTTTTTT	316	279.153	329	284.388	1.15687	47.9416	39.1778
TGAAAAAAAAAAA	199	164.533	199	167.263	1.18974	34.5735	37.8481
AAAGAAGAAGAAA	92	61.2258	95	62.1232	1.52922	40.352	37.4642
GTTTTTTTTTTTC	99	68.0435	100	69.0494	1.44824	37.0348	37.1223
TAAAAATAAAAAT	140	107.89	151	109.565	1.37818	48.4349	36.4747
AGTTTTTTTTTTT	473	438.173	480	447.711	1.07212	33.4268	36.1755
CTTTTTTTTTTTG	171	139.032	172	141.272	1.21751	33.851	35.3904
GTTTTTTTTTTGT	139	108.652	140	110.341	1.26879	33.329	34.2384

Table A.78: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAAAATAAAAA	207	168.305	216	171.303	1.26093	50.0789	42.8363
TAAAAAAAAAAAT	93	58.9379	93	59.8666	1.55345	40.9647	42.4187
CTTTTTTTTTTTG	120	89.6925	120	91.1575	1.3164	32.9884	34.9325
TATAAAATAAAAT	59	33.3038	59	33.8127	1.74491	32.8454	33.74
ATTTTTTTTTTTA	80	55.3225	80	56.1905	1.42373	28.2624	29.5078
ATTTAAAAAAAAA	122	96.3067	122	97.8916	1.24628	26.8596	28.851
CAAAAAAAAAAAC	112	86.6319	112	88.0418	1.27212	26.9569	28.7651
GTTTTTTTTTGTT	86	61.6851	86	62.6602	1.37248	27.2294	28.5783
AAAGAAAAAAAAA	296	269.083	300	274.386	1.09335	26.7742	28.221
TTTTAAAAAAAAT	63	42.1068	64	42.757	1.49683	25.8144	25.3843
TATAAAAATAAAT	44	24.7398	44	25.1138	1.75202	24.6739	25.3341
GACAAAAAAAAAA	176	153.524	177	156.216	1.13305	22.1094	24.046
ACAAAAAAAAAAC	83	62.1328	84	63.1155	1.33089	24.0114	24.034
GTATATATATATAC	18	4.83159	19	4.90283	3.87531	25.7379	23.6735
ATTTTTTTTTATTT	60	40.6319	60	41.2583	1.45425	22.4696	23.3874
CTTACAAAATATTG	23	8.38664	23	8.51087	2.70243	22.8655	23.2036
TTTTTTTTTTCTTT	297	274.725	301	280.169	1.07435	21.5869	23.1542
TCAAAAAAAAAAA	245	223.464	248	227.676	1.08927	21.2051	22.5414
TAAAATAAAAATAT	55	36.6783	55	37.2409	1.47687	21.4458	22.2832
TGAAAAAAAAAAA	139	118.78	139	120.785	1.15081	19.5244	21.8509
TTAATAAAATAAAT	58	40.3189	58	40.9402	1.4167	20.2031	21.09
GTTTTTTTTTGTTT	69	51.0109	69	51.8071	1.33186	19.774	20.8426
TTTTTATTTTTTTT	160	140.641	161	143.072	1.12531	19.0069	20.6345
TTTTTTTTTTGTTA	133	113.923	133	115.835	1.14818	18.3782	20.5924
TTTTTTTAAAAAAA	123	104.078	123	105.806	1.16251	18.5211	20.5465

Table A.79: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAAA	607	575.36	617	590.722	1.04449	26.8543	32.4944
TTTTTTTTTTTTTTG	589	562.724	597	577.612	1.03357	19.7096	26.8797
TTTTTTTTTTTTGTC	125	101.049	125	102.837	1.21551	24.3959	26.5882
AAAAAAAATAAAAAA	145	120.755	152	122.936	1.23641	32.2567	26.5307
AGTTTTTTTTTTTTT	257	234.568	259	239.307	1.08229	20.4815	23.4725
TTTTAAAAAAAAAAT	35	18.4014	35	18.6985	1.87181	21.9417	22.5023
AAAAAAAAAAAAAAC	396	374.686	403	383.252	1.05153	20.2479	21.9091
TAAAAAAAAAAAAAT	56	38.2932	56	38.9257	1.43864	20.367	21.2844
GTTTTTTTTTTTTTT	418	397.27	425	406.524	1.04545	18.8897	21.2613
AAAAAAAAAAAAACT	221	201.784	224	205.736	1.08877	19.0512	20.1033
TAAAAAAAAAAAATA	31	16.5745	31	16.8415	1.84069	18.9143	19.4098
CTCTTTTTTTTTTTT	115	97.8413	116	99.5665	1.16505	17.7207	18.5823
TGTTTTTTTTTTTTG	47	32.2812	47	32.8107	1.43246	16.8914	17.6562
CTTTTTTTTTTTGTT	47	32.3912	47	32.9226	1.42759	16.7315	17.4963
AAAAAAGAAAAAAAA	144	127.55	144	129.87	1.1088	14.8721	17.468
TAACAAAAAAAAAAA	92	76.1729	94	77.4851	1.21314	18.1616	17.368
CATTATTAATTTATA	30	16.8524	31	17.124	1.81033	18.3988	17.3012
TATAAAATAAAAATA	48	33.5866	48	34.1384	1.40604	16.3574	17.1396
AAAAAAAACAAAAAA	94	78.6015	98	79.9591	1.22563	19.9383	16.817
TTTTTTATTTTTTTT	100	84.6559	101	86.1277	1.17268	16.0883	16.6576
CAAAAAAAAAAGAAA	42	28.3214	42	28.7839	1.45915	15.8698	16.5502
CAAAAAAAAAAAACA	56	41.7161	56	42.4078	1.32051	15.5691	16.4901
GATTTTTAATATTTC	21	9.70208	21	9.85714	2.13043	15.8828	16.2158
TTCTTTTTTTTTTTG	47	33.3323	47	33.8798	1.38726	15.3845	16.1501
TTTTTTTTTGTTTTT	121	106.201	122	108.091	1.12868	14.7682	15.7849

Table A.80: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTTTTG	462	438.888	466	449.967	1.03563	16.3154	23.7102
ATTTTTTTTTTTTTA	42	28.453	42	28.9504	1.45076	15.6276	16.3554
TTAAACTATAAAATAT	14	4.80698	14	4.88889	2.86364	14.7293	14.9658
GAGATCAACATTAATC	10	2.28673	10	2.32558	4.3	14.5862	14.7546
ACTTTTTTTTTTTTTT	130	116.689	132	118.922	1.10997	13.7725	14.0433
TAAACTTTTTTTTTA	18	8.30689	18	8.44898	2.13043	13.6139	13.9191
AAAAAAAAAAAAAATC	150	136.997	150	139.671	1.07395	10.7017	13.6013
CTTTTTTTTTTTTTTT	312	299.249	314	306.008	1.02612	8.09585	13.0187
TTTTTTTATTTTTTTT	73	61.4391	74	62.5511	1.18303	12.438	12.5861
TTTTTTTTTTTTTTTC	202	189.811	203	193.704	1.04799	9.51522	12.5727
AAAGAAGAAGAAGAAA	20	10.7214	20	10.9052	1.83398	12.1298	12.4699
TTAAAAAAAAAAAAAA	208	195.908	208	199.949	1.04026	8.21087	12.4581
CAAAAAAAAAAAAAGG	20	10.7668	21	10.9515	1.91755	13.672	12.3853
AAAAAAAAAAAAACAG	49	38.2259	49	38.9011	1.2596	11.3091	12.1671
AATATATATATATATA	282	270.199	285	276.152	1.03204	8.98814	12.0555
CTTTTTCTTCTTCTTC	20	10.988	20	11.1765	1.78947	11.6384	11.9786
AAAAAAGAAAAAAAAA	105	93.8601	105	95.616	1.09814	9.83012	11.7763
AAAAAAAGAAAAAAAA	103	91.8845	103	93.6	1.10043	9.85696	11.7623
GAAAAAAAAGAAAAAT	11	3.79185	11	3.85638	2.85241	11.5298	11.7155
ATATTTTTATTTTATA	20	11.1767	20	11.3684	1.75926	11.2979	11.6381
AAAAAAAAATAAAAAA	84	73.2506	85	74.5926	1.13952	11.1019	11.5022
TATAAAATAAAAATAT	30	20.514	30	20.8696	1.4375	10.8872	11.4027
GTAAACTATAAAATAA	42	32.1033	42	32.6667	1.28571	10.5552	11.2859
CAAAAAAAAAAAACAA	26	16.9383	26	17.2308	1.50893	10.6964	11.1415
TATATATATATATATT	303	292.21	310	298.771	1.03758	11.4374	10.9863

Table A.81: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAAAAAAAAAAAAAAA	360	338.301	364	346.584	1.05025	17.8464	22.3804
TAAAAAATTAATATGAT	22	10.191	22	10.3774	2.12	16.5312	16.9299
TTTTTTTTTTTTTTTG	357	340.979	360	349.345	1.0305	10.816	16.3917
AAAAAAATTAATATGAC	27	15.5073	27	15.7925	1.70968	14.4802	14.9722
CTTTTTTTTTTTTTTT	244	230.231	245	235.395	1.0408	9.79786	14.173
AAAACTTTTATATATAA	11	3.16209	11	3.21951	3.41667	13.5153	13.7133
AATATATATATATATAT	268	255.344	271	261.194	1.03754	9.98776	12.9641
GTTTTTTTTTTTTTTT	248	236.071	251	241.393	1.0398	9.79588	12.2255
TATATATATATATATAA	187	175.386	190	179.139	1.06063	11.1838	11.9899
CTTTTAAAAAATAAATC	19	10.3114	19	10.5	1.80952	11.2682	11.6126
AAAAAAAAAAAAAAAG	250	238.939	251	244.338	1.02727	6.75203	11.3137
TCTTTTTTTTTTTTTT	144	133.37	144	136.118	1.0579	8.10568	11.0425
CTCTTTTTTTTTTTTT	67	57.1099	67	58.2047	1.15111	9.42865	10.7009
TAAAAAAAAAAAAAAT	29	20.3113	29	20.6867	1.40187	9.79637	10.3274
TTTCTTCTTCTTCTTCC	13	5.9909	13	6.1	2.13115	9.83659	10.0712
AAAAAAAAAAAAAAATC	120	110.368	120	112.594	1.06577	7.64391	10.0403
AAAAATAAAATAAAAAA	39	30.1608	40	30.7238	1.30192	10.5537	10.0237
ATCTCTCTCTCTCTCTG	7	1.71992	7	1.7511	3.99748	9.69964	9.82543
TCAAAAAAAAAAAAAA	120	110.591	121	112.822	1.07249	8.46778	9.79884
TTTGTTGTTGTTGTTGA	6	1.18712	6	1.20863	4.96429	9.61362	9.72137
ATAATTAATTAAATTAT	7	1.8093	7	1.84211	3.8	9.34501	9.4708
TAAACAAAAAAAAAAG	9	3.15696	9	3.21429	2.8	9.26657	9.42854
GTTTTTATTTTATTTTC	9	3.26569	9	3.325	2.70677	8.96179	9.12377
ACTTTTTTTTTTTTTA	15	8.30166	15	8.4532	1.77448	8.60257	8.87393
TTTTTTTTTGTTTTTT	44	35.9883	44	36.6639	1.20009	8.02543	8.84381

Table A.82: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AATATATATATATATATA	253	233.186	255	238.7	1.06828	16.8439	20.6326
TATATATATATATATATT	261	245.261	265	251.117	1.05529	14.2603	16.2337
AATCAACATTAATAAAAA	14	5.35969	14	5.46341	2.5625	13.1738	13.4421
AATTTTTTTTTTTTTTTT	83	72.2073	83	73.6954	1.12626	9.86876	11.5618
GATCAACATTAATAAAAT	23	14.3061	23	14.5854	1.57692	10.4759	10.9206
CCAAAAAAAAAAAAAAAG	17	9.26045	17	9.44035	1.80078	9.99975	10.3268
TGAGAGAGAGAGAGAGAA	7	1.64547	7	1.6772	4.17361	10.0015	10.1352
CCTCTCTCTCTCTCTCTT	7	1.84017	7	1.87567	3.73201	9.21862	9.35235
AAAAAAAAAAAAAAAATT	83	74.3047	84	75.8389	1.10761	8.58523	9.18527
GAAAAAAAAAAAAAACAG	8	2.77403	8	2.82759	2.82927	8.32014	8.47311
TCTTTTTTTTTTTTTTTT	118	110.011	118	112.357	1.05023	5.78272	8.27196
ACCAAAAAAAAAAAAAAC	17	10.4819	18	10.6857	1.68449	9.38635	8.22065
CTTTTTTTTTTTTTTTTT	187	179.068	188	183.119	1.02665	4.94548	8.10509
GCTCTCTCTCTCTCTCTG	4	0.533245	4	0.543517	7.35948	7.98396	8.06027
GAGAAGAAGAAGAAGAAA	11	5.36433	11	5.46814	2.01165	7.68852	7.89937
ACAATATGACCCTTATAT	4	0.560629	4	0.571429	7	7.78364	7.85996
TAAAAAAAAAAAAAAAAT	21	14.462	21	14.7444	1.42427	7.42686	7.83291
GATTTTTTTTTTTTTTG	20	13.547	20	13.8113	1.44809	7.40496	7.79134
CTATATATATATATATAC	6	1.72024	6	1.75341	3.42189	7.38117	7.49577
TATATATATATATATATG	128	120.772	129	123.371	1.04563	5.75563	7.4405
ACAAAAAAAAAAAAAAAA	127	119.81	128	122.386	1.04587	5.74071	7.40183
GATTTTTTTTTTTTTGT	10	4.82558	10	4.91892	2.03297	7.09496	7.28655
TTTTTTTTTTTTTTTAA	119	111.949	119	114.34	1.04076	4.75409	7.26868
GGAAAAAAAAAAAAAAG	7	2.48666	7	2.53465	2.76172	7.11097	7.24478
CGATCCGAACTGCTCTTT	4	0.654066	4	0.666667	6	7.16704	7.24336

Table A.83: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTTTTTTTC	18	9.52818	18	9.72433	1.85103	11.0833	11.4501
CAAAAAAAAAAAAAAAAC	29	21.0285	29	21.466	1.35098	8.72398	9.32107
AAAAAAAAAAAAAAAAACT	82	73.4139	82	75.0135	1.09314	7.30225	9.06965
TAGAAGAAGAAGAAGAAGC	4	0.516844	4	0.527397	7.58442	8.10438	8.18523
TTTATTATTATTATTATTT	6	1.55643	6	1.58824	3.77778	7.97482	8.0962
CAAAAAAAAAAAAAAGTT	11	5.40012	11	5.51087	1.99606	7.6029	7.82622
TTAAAAAAAAAAAAAAAA	79	72.103	79	73.6721	1.07232	5.51604	7.21687
TTTTTTTTTTTTTTTGGG	28	21.6595	28	22.1103	1.26638	6.61251	7.18934
TTTTTTTTTTTTTTTGTC	48	41.3424	48	42.2182	1.13695	6.1608	7.16703
TTTTTTTTTTTTTTTTGG	78	71.2016	78	72.75	1.07216	5.43503	7.11305
CGAGAGAGAGAGAGAGAGC	3	0.29743	3	0.303502	9.88462	6.87294	6.93356
TATGATGATGATGATGATT	3	0.297971	3	0.304054	9.86667	6.86749	6.92811
CAAAAAGAAAAAAAAAAAC	5	1.37197	5	1.4	3.57143	6.36483	6.46597
ACTCTCTCTCTCTCTCTCC	3	0.354466	3	0.361702	8.29412	6.34664	6.40727
ATCAAAAAAAAAAAAAAAG	5	1.40922	5	1.43802	3.47701	6.23087	6.33201
AATTTTTTTTTTTTTTTTT	66	59.9885	66	61.2804	1.07702	4.89688	6.3031
AAAAAAAAAAAAAAAAAGT	56	50.1329	56	51.2032	1.09368	5.01479	6.19779
ACATATATATATATATATA	77	71.0983	79	72.6443	1.08749	6.62597	6.14011
AAAAAAAAAAAAAAAAAAT	124	118.033	124	120.704	1.0273	3.34016	6.11483
TTTTTTTTTTTTTTTTCT	50	44.2653	50	45.2055	1.10606	5.04023	6.09108
TCATTTTTTTTTTTTTTG	7	2.99531	7	3.0566	2.29012	5.80024	5.94204
CTTTTTTTTTTTTTTTTT	143	137.22	143	140.375	1.0187	2.64988	5.90033
TATATATATATATATATGT	80	74.3157	80	75.9362	1.05352	4.17068	5.89631
GAAAAAAAAAAAAAAACG	4	0.91846	4	0.93722	4.26794	5.80453	5.88541
CTTTTTCTTTTTCTTTTCC	4	0.922337	4	0.941176	4.25	5.78768	5.86855

Table A.84: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AATATATATATATATATATA	210	194.533	211	199.442	1.05795	11.8863	16.0664
TATATATATATATATATATT	221	206.19	223	211.439	1.05468	11.8716	15.33
TTTTTTTTTTTTTTTTTTTG	162	147.616	163	151.211	1.07797	12.2374	15.0626
CAAAAAAAAAAAAAAAAAAA	163	150.53	163	154.203	1.05705	9.04312	12.9732
TTTTTTTTTTTTTTTTTTGT	80	68.7542	80	70.3259	1.13756	10.3109	12.1191
TAATTTAGATATCAAATCTA	16	7.73451	17	7.90244	2.15123	13.0227	11.6303
AAATTTAGATATCAAATCTT	21	12.0782	22	12.3415	1.78261	12.7177	11.6155
ACGTTACTTAATAACAATTG	23	14.0054	23	14.3111	1.60714	10.9125	11.4092
GCGTTACTTAATAACAATTA	12	4.95925	12	5.06667	2.36842	10.3467	10.6038
ATTTTTTTTTTTTTTTTTTA	16	8.44293	16	8.62635	1.85478	9.88427	10.2281
GTTTTTTTTTTTTTTTTTTT	104	95.0928	105	97.3138	1.07898	7.98204	9.31192
AAATATATATATATATATAT	109	101.339	109	103.718	1.05093	5.41464	7.94375
CGAGAGAGAGAGAGAGAGAC	3	0.216604	3	0.221277	13.5577	7.82086	7.88489
TATATATATATATATATATC	63	56.1574	63	57.4278	1.09703	5.83416	7.24349
AATTTTTTTTTTTTTTTTTA	10	4.90639	10	5.01266	1.99495	6.90619	7.12046
CATATATATATATATATATG	9	4.11063	9	4.1996	2.14306	6.86013	7.05284
CTTTTTTTTTTTTTTTTTTT	111	104.584	111	107.045	1.03695	4.02698	6.60914
AAAAAAAAAAAAAAAAAAAG	111	104.644	111	107.107	1.03635	3.9633	6.54558
ATGTTTTTTTTTTTTTTTTG	7	2.77779	7	2.83784	2.46667	6.32007	6.46979
AAAAAAAAAAAAAAAAAAAC	90	83.8955	90	85.8372	1.0485	4.26212	6.32144
GATTCAAAAAAAAAAAAAG	3	0.376493	3	0.384615	7.8	6.16237	6.2264
ATTTTTTTTTTTTTTTTTGA	9	4.64026	9	4.74074	1.89844	5.76928	5.96208
ATGAATAACACAAAACTTTA	6	2.28398	6	2.33333	2.57143	5.66677	5.79504
GTTTTTTTTTTTTTTGAAG	3	0.46065	3	0.470588	6.375	5.55715	5.62119
GGTTGTTGTTGTTGTTGTTA	2	0.128801	2	0.131579	15.2	5.44259	5.48527

A.5 The *Arabidopsis thaliana* Distal Promoters

A.5.1 Timing Information

Table A.85: Full Size and Timing Information, Distal Promoters

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	6.136547	7.587327	95.409623	13.723874
2	0	0.000001	11.504371	7.648788	168.929291	19.153159
3	1	0.000003	17.786348	7.876829	93.132696	25.663177
4	2	0.000012	27.409133	7.446788	39.534179	34.855921
5	3	0.000046	36.493901	9.065967	6.309500	45.559868
6	4	0.000186	46.499415	7.982420	3.748664	54.481835
7	5	0.000743	59.418827	8.906611	2.350214	68.325438
8	6	0.002971	74.714318	23.299167	1.878490	98.013485
9	7	0.011883	99.730552	57.102017	2.997452	156.832569
10	8	0.047409	128.083779	224.766201	8.761543	352.84998
11	9	0.183115	159.094246	793.066140	31.567568	952.160386
12	10	0.622073	193.648582	1617.926127	99.224997	1811.574709
13	11	1.686000	235.329202	3749.135607	225.043931	3984.464809
14	12	3.566710	269.500937	5067.892382	369.668750	5337.393319
15	13	6.128855	312.903055	6539.620762	490.762339	6852.523817
16	14	9.084096	347.534894	12236.466104	587.315891	12584.000998
17	15	12.219183	384.540443	11896.158787	667.592003	12280.69923
18	16	15.429060	430.605791	7777.418279	739.966548	8208.02407
19	17	18.670213	464.859145	7448.516166	811.136075	7913.375311
20	18	21.925547	512.117295	7747.970193	883.128174	8260.087488

A.5.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.86: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
C	27168	27168	9590626	9.59063e+06	1	0	0
G	27168	27168	9709601	9.7096e+06	1	0	0
T	27168	27168	17556578	1.75566e+07	1	0	0
A	27168	27168	17473994	1.7474e+07	1	0	0

Table A.87: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CG	27168	27168	1256758	1.71397e+06	0.733246	-389940	0.0346982
AG	27168	27167.9	3178819	3.12282e+06	1.01793	56494.4	0.0609948
CC	27168	27168	1750497	1.69296e+06	1.03398	58499.3	0.0344554
CA	27167	27167.9	3413706	3.08456e+06	1.10671	346115	-0.939779
GC	27167	27168	1588128	1.71397e+06	0.926581	-121101	-0.965285
AT	27168	27167.9	5124500	5.64659e+06	0.90754	-497170	0.115984
AC	27168	27167.9	2821876	3.08456e+06	0.914839	-251166	0.0602044
GA	27168	27167.9	3454108	3.12282e+06	1.10608	348266	0.0609948
CT	27168	27167.9	3165027	3.09914e+06	1.02126	66585.9	0.0605053
GG	27167	27168	1806863	1.73523e+06	1.04128	73093.2	-0.965026
GT	27167	27167.9	2855806	3.13758e+06	0.910193	-268727	-0.938684
TA	27168	27167.9	4257679	5.64659e+06	0.754027	-1.20206e+06	0.115984
TC	27167	27167.9	3425297	3.09914e+06	1.10524	342751	-0.939478
AA	27168	27167.9	6339828	5.62003e+06	1.12808	764047	0.115376
TG	27167	27167.9	3462164	3.13758e+06	1.10345	340820	-0.938684
TT	27167	27167.9	6402543	5.67327e+06	1.12854	774254	-0.88339

Table A.88: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CGA	27168	27167.8	433721	447081	0.970117	-13158.3	0.243002
ACC	27168	27167.8	554736	515053	1.07705	41173.4	0.190487
GAC	27168	27167.8	492378	557804	0.882708	-61429.6	0.16595
TAC	27168	27167.9	689528	687573	1.00284	1958.02	0.11512
CTA	27168	27167.9	710992	767557	0.926306	-54427.3	0.0955636
TAG	27168	27167.9	705399	774545	0.910727	-65963.2	0.0941609
AAA	27168	27167.9	2427774	2.30019e+06	1.05547	131064	0.0906292
AAC	27168	27167.9	1066639	1.02382e+06	1.04182	43703.1	0.0652509
TAA	27168	27167.9	1411234	1.54475e+06	0.913568	-127572	0.0634345
AGA	27168	27167.9	1247334	1.13084e+06	1.10302	122301	0.0609803
ATA	27168	27167.9	1432516	1.24275e+06	1.1527	203566	0.0594104
TAT	27168	27167.9	1449270	1.24863e+06	1.16069	215965	0.0593901
GAA	27168	27167.9	1220985	1.2532e+06	0.974292	-31799.6	0.0593781
TCG	27167	27167.8	431278	448852	0.960847	-17225.2	-0.758661
AGC	27167	27167.8	562437	519936	1.08174	44192.5	-0.812568
CAC	27167	27167.8	571867	551280	1.03734	20967.2	-0.830609
GTC	27167	27167.8	491842	557169	0.882752	-61338.3	-0.833708
GTG	27167	27167.8	586998	563166	1.04232	24329.2	-0.836778
CCT	27167	27167.8	522506	577686	0.904481	-52456.5	-0.843868
CTC	27167	27167.9	724414	617498	1.17314	115679	-0.861073
CCA	27167	27167.9	688722	623075	1.10536	68989.5	-0.863248
GAG	27167	27167.9	735731	628361	1.17087	116061	-0.86526
GGA	27167	27167.9	646730	642776	1.00615	3965.99	-0.870513
TGG	27167	27167.9	715872	644275	1.11113	75435.2	-0.87104
GTA	27167	27167.9	689559	692567	0.995657	-3001.38	-0.886298

Table A.89: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CGCC	19896	19462.9	47053	34278.7	1.37266	14904.1	437.887
GGCG	20077	19779.8	48606	35421	1.37224	15381	299.383
CGAC	24515	24370.3	75105	61826.3	1.21477	14612.3	145.179
GACG	24773	24697.9	82816	65212.9	1.26993	19790	75.2165
GTCG	24429	24380.7	75462	61927.7	1.21855	14916	48.3862
CGTC	24584	24581.1	80391	63956.9	1.25696	18384.9	2.88711
GAAA	27168	27167.8	454108	467564	0.971221	-13260.5	0.235496
GATT	27167	27167.4	336882	325745	1.03419	11325.3	-0.432665
TTCA	27167	27167.7	387003	397111	0.974545	-9978.63	-0.687055
AAAC	27167	27167.7	432914	408459	1.05987	25173.1	-0.703605
AAAG	27167	27167.8	415822	454344	0.915214	-36840.5	-0.753288
AAGA	27167	27167.8	483898	465555	1.0394	18699.9	-0.762854
TTTC	27167	27167.8	454596	467700	0.971982	-12918.8	-0.764604
TTAT	27167	27167.8	476206	484198	0.983495	-7925.53	-0.777317
TTTG	27167	27167.8	513030	499357	1.02738	13858.1	-0.787968
ATTT	27167	27167.9	642312	641019	1.00202	1294.78	-0.856398
TTTT	27167	27167.9	963707	950531	1.01386	13267.4	-0.910542
ACTT	27166	27167.3	299111	310658	0.962832	-11329.3	-1.28152
TGAT	27166	27167.4	368922	326678	1.12931	44864.4	-1.43986
TCAT	27166	27167.5	344424	332783	1.03498	11842.4	-1.48252
AATG	27166	27167.5	324120	333481	0.97193	-9228.21	-1.48693
ATGA	27166	27167.5	346065	335282	1.03216	10954.5	-1.49791
ATTG	27166	27167.5	311234	336756	0.924211	-24529.9	-1.50648
ATCT	27166	27167.6	333078	361013	0.922619	-26825.7	-1.60832
AGAT	27166	27167.6	332947	362337	0.918889	-28164.1	-1.61231

Table A.90: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CGCCG	8419	6989.03	14049	8095.41	1.73543	7744.56	1567.19
CGGCG	8397	7054.94	14104	8184.46	1.72327	7675.69	1462.31
TAGGG	14576	13422.7	22134	18545.1	1.19352	3915.71	1201.53
CGACG	11212	10087.2	17888	12632.4	1.41605	6222.67	1185.25
CGTCG	10949	9899.09	17501	12334.2	1.4189	6123.32	1103.72
CCCTA	14423	13442.7	22044	18584.7	1.18614	3762.92	1015.25
TACCT	20021	19351.8	39253	33906.2	1.15769	5747.8	680.68
ACTCG	15576	14970.8	24378	21797.1	1.11841	2728.02	617.245
GAATC	24571	23969.4	71637	58214.8	1.23056	14862.7	609.081
TGCGT	13340	12745.4	19429	17236.2	1.12722	2326.76	608.253
GGCCC	7425	6848.54	9569	7906.57	1.21026	1826.1	600.064
CGAGT	15643	15071.3	24677	22022.2	1.12055	2808.78	582.411
ACGCA	13274	12704.9	19139	17159.9	1.11533	2089.09	581.628
GATAC	20413	19855.1	38719	35717.3	1.08404	3124.46	565.684
AGGTA	20357	19806.9	41603	35538.7	1.17064	6554.56	557.643
GTAAG	21212	20670.5	43181	38933.9	1.10908	4470.73	548.548
GTTAC	21260	20730.8	43236	39187.8	1.1033	4250.5	535.879
GGTCC	12761	12237.9	18227	16294.9	1.11857	2042.35	534.147
GTATC	20396	19868.8	38887	35768.5	1.08719	3250.68	534.104
GTGCC	10118	9599.66	13233	11866.3	1.11518	1442.55	532.087
GATTC	24515	23989.7	71556	58387.8	1.22553	14552.6	531.031
TTAGG	21439	20921	44823	40003.6	1.12047	5098.73	524.388
GGGCC	7429	6932.83	9489	8019.71	1.18321	1596.34	513.515
ATTCG	19615	19113.8	36640	33090.4	1.10727	3733.57	507.665
GTACG	11332	10851.6	15930	13878.5	1.14782	2196.22	490.844

Table A.91: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAATTG	13193	11983	18664	15842.1	1.17813	3059.51	1269.17
CATATC	12451	11267.2	17274	14588	1.18413	2919.41	1243.89
CAATTC	13120	11934.7	18603	15755.8	1.18071	3090.25	1242.25
TAGAAT	14113	12946	20711	17626.2	1.17501	3340.25	1218.07
GATAAG	11231	10112.8	15126	12679.5	1.19295	2668.67	1177.84
GCTAAG	7140	6061.87	8549	6876.09	1.24329	1861.66	1168.78
CTTATC	11378	10283	15287	12952.6	1.18023	2533.19	1151.34
GATATG	12673	11610.4	17920	15182	1.18034	2971.21	1109.83
TAGGGT	7310	6280.99	9084	7160.31	1.26866	2161.65	1109.04
ACCCTA	7333	6327.09	9314	7220.48	1.28994	2371.32	1081.94
TAGGAT	8967	7954.28	11210	9434.31	1.18822	1933.2	1074.61
CATTAC	10681	9685.45	13897	12005.5	1.15755	2033.19	1045.05
TAGCTA	10216	9229.8	13386	11304.9	1.18409	2261.87	1037.1
GTTTAG	13291	12296.9	19029	16411	1.15953	2816.54	1033.22
GAAATG	15641	14689.3	24112	21186.7	1.13807	3118.54	981.883
ATTCTA	14211	13285.3	21204	18283.7	1.15972	3142.02	957.205
ACTTCA	13960	13052.6	20258	17831	1.13611	2585.19	938.27
GCAATG	9596	8702.64	12329	10516.2	1.17238	1960.8	937.719
GATATC	11059	10164.9	14606	12762.8	1.14442	1970.27	932.269
CTTAGC	6855	5984.13	8247	6775.97	1.21709	1620.26	931.373
GTAATG	10667	9782.29	13944	12156.8	1.14701	1912.56	923.561
CAAAAC	20065	19165.7	40095	33282.9	1.20467	7466.02	920.04
ATCCTA	8879	8034.64	11086	9548.43	1.16103	1655.21	887.253
CATTTC	15426	14570	23547	20927.6	1.12516	2776.86	880.676
CATTGC	9480	8642.8	12178	10428.1	1.16781	1889.17	876.499

Table A.92: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTG	10688	8475.44	14445	10188.3	1.41781	5042.92	2479.07
CTTTTTC	9890	7847.49	12681	9288.03	1.36531	3948.59	2287.85
CAAAAAC	10259	8217.42	13765	9814.76	1.40248	4655.9	2276.46
GAAAAAG	9761	7790.54	12591	9207.82	1.36742	3940.09	2200.96
GTTCTTG	6425	4930.19	7728	5456.42	1.41631	2689.78	1701.46
CAAGAAC	6270	4783.81	7602	5277.65	1.44041	2774.2	1696.28
GAAAATG	8919	7514.75	11163	8822.79	1.26525	2626.29	1527.97
CATTTTC	8722	7355.56	10812	8602.99	1.25677	2471.05	1486.15
GATTTTG	9804	8479.91	12764	10194.8	1.25201	2868.75	1422.47
CAAAATC	9589	8271.19	12416	9892.17	1.25513	2821.44	1417.62
GATGAAG	6815	5560.76	8396	6240.2	1.34547	2491.45	1386.11
GAAGATG	7711	6509.04	9768	7463.01	1.30885	2629.08	1306.68
GAAATTG	6961	5778.14	8183	6515.7	1.25589	1864.45	1296.43
CTGTTTC	5726	4571.97	6715	5020.99	1.33739	1952.16	1288.77
GATAAAG	5202	4075.99	5847	4429.38	1.32005	1623.54	1268.92
CTTCATC	6540	5396.39	8105	6033.71	1.34329	2391.94	1257.04
GATAATG	4925	3817	5520	4125.48	1.33803	1607.4	1255.18
CATCTTC	7393	6240.63	9380	7111.3	1.31903	2597.27	1252.76
GAATTTG	7341	6199.2	8795	7057.41	1.24621	1935.82	1241.03
CAATTTC	6863	5736.09	7995	6462.19	1.2372	1701.71	1230.99
CATAAAC	5837	4729.47	6801	5211.58	1.30498	1810.33	1228.13
GTTTATG	6032	4931.14	7061	5457.58	1.2938	1818.78	1215.5
GAACAAG	5394	4316.84	6213	4715.07	1.31769	1714.05	1201.59
CTTTATC	5169	4105.79	5852	4464.57	1.31077	1583.62	1190.32
GAAACAG	5646	4592.01	6661	5045.17	1.32027	1850.68	1166.63

Table A.93: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTA	5809	4887.98	7224	5407.46	1.33593	2092.29	1002.81
TAAAAAAT	5897	5014.08	7353	5562.19	1.32196	2052.34	956.455
GAAAAAAG	3583	2836.49	3925	3006.2	1.30563	1046.75	837.104
TTATATAA	4822	4128.99	5704	4494.25	1.26918	1359.66	748.158
CTTTTTTC	3551	2888.7	3907	3064.76	1.27481	948.619	733.008
AATATATT	5457	4918.71	6744	5445.09	1.23855	1442.81	566.722
CAAGAAAC	2935	2480.49	3218	2610.21	1.23285	673.618	493.816
GAAAAATG	3191	2752.37	3454	2912.11	1.18608	589.44	471.86
GTTTCTTG	2931	2501.51	3199	2633.43	1.21477	622.368	464.418
GTTTTTGA	3531	3110.79	3842	3315.28	1.15888	566.502	447.394
GAAAAAAC	3028	2618.45	3255	2762.99	1.17807	533.429	440.028
CAATTTTT	4470	4054.76	5004	4406.56	1.13558	636.231	435.806
TCAAAAAC	3439	3032.33	3715	3226.52	1.15139	523.718	432.791
ATTTTGTA	4114	3706.2	4643	3998.5	1.16119	693.858	429.455
CTTTATTC	1631	1257.91	1688	1292.48	1.30602	450.667	423.633
GAAGAAAG	3863	3464.96	4311	3719.61	1.15899	636.09	420.079
GTTTTATG	2179	1798.17	2299	1866.95	1.23142	478.58	418.578
GTTTTAAG	1963	1589.13	2060	1643.23	1.25363	465.647	414.76
ATTTTTCA	4116	3722.26	4532	4017.17	1.12816	546.494	413.864
TAGAAAAT	3590	3204.71	3925	3421.93	1.14701	538.363	407.573
TAAGAAGT	1472	1118.55	1524	1146.23	1.32958	434.129	404.197
CAAAAAAG	3406	3024.87	3710	3218.09	1.15286	527.727	404.196
CTTGTTTC	2358	1989.48	2509	2073.31	1.21014	478.555	400.715
GATATATC	1380	1040.23	1429	1064.38	1.34256	420.957	390.05
AAAAATTG	4443	4074.6	5041	4429.96	1.13793	651.37	384.576

Table A.94: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAAAAAAAT	3076	2308.15	3469	2421.77	1.43242	1246.65	883.38
ATTTTTTTA	3015	2250.24	3379	2358.31	1.43281	1215.2	882.074
TTATATATT	2411	1775.93	2629	1843.98	1.42572	932.441	737.087
AATATATAA	2307	1701.08	2496	1763.7	1.41521	866.797	702.904
ATTTTCTTA	1999	1545.04	2118	1597.08	1.32617	597.899	514.937
TAAGAAAAT	1910	1530.88	2033	1582.01	1.28507	509.911	422.611
CTTTTTTTC	1662	1320.26	1738	1358.84	1.27903	427.731	382.583
TAAAATTTT	2519	2167.68	2782	2268.09	1.22658	568.177	378.362
ATTTTTTTC	2326	1985.14	2483	2069.65	1.19972	452.128	368.583
GAAAAAAAG	1573	1251.14	1651	1285.99	1.28383	412.503	360.111
GATATATAA	1032	731.496	1059	744.498	1.42244	373.161	355.176
TTCTCTCTT	1775	1455.72	1872	1502.16	1.2462	412.032	351.985
TTATATATC	1062	765.939	1102	780.059	1.41271	380.754	347.068
GTATATATG	978	691.052	1005	702.8	1.42999	359.459	339.654
GAAAAAAAC	1551	1254.03	1611	1289.04	1.24977	359.187	329.646
TAAATTTTT	2224	1927.86	2462	2007.68	1.22629	502.24	317.807
CTCTTTCTC	1368	1094.53	1478	1121.67	1.31768	407.744	305.098
GAAAAAAAT	2209	1929.9	2347	2009.88	1.16773	363.93	298.375
TAGAAAAAT	1354	1088.54	1411	1115.4	1.26502	331.712	295.484
ATTTTTTCA	1450	1188.85	1496	1220.52	1.22571	304.462	287.932
TCAAAAAAT	1205	950.611	1231	971.52	1.26709	291.403	285.741
ATTTTTCTA	1372	1116.95	1449	1145.13	1.26536	341.034	282.18
GTTTTTTTC	1594	1336.63	1655	1376.12	1.20266	305.402	280.696
TTTTTCTTT	5970	5699.95	7197	6422.73	1.12055	819.168	276.346
TTTTTGATA	1506	1254.48	1591	1289.52	1.2338	334.262	275.195

Table A.95: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTA	1520	1037.56	1606	1062.67	1.51129	663.223	580.393
AATATATATT	1166	724.697	1229	737.855	1.66564	627.047	554.526
TAAAAAAAAT	1475	1039.15	1555	1064.33	1.46102	589.551	516.624
TTATATATAA	944	569.023	974	577.663	1.6861	508.837	477.859
AAAAAAAATC	1922	1645.05	2029	1704.61	1.1903	353.47	299.059
CATATATATG	556	325.781	568	329.231	1.72523	309.766	297.206
AAAGAAGAAA	1311	1056.67	1350	1082.63	1.24696	297.957	282.738
TTTCTTCTTT	1391	1147.21	1457	1177.42	1.23745	310.42	268.036
TTATATATAC	576	370.299	590	374.53	1.57531	268.125	254.475
TTATATATAG	511	324.5	517	327.929	1.57656	235.363	232.037
GATTTTTTTT	1944	1727.37	2059	1792.77	1.1485	285.084	229.679
TTTTTAAAAA	2371	2153.22	2621	2253.44	1.16311	396.023	228.44
CATATATATC	358	189.359	364	190.881	1.90695	234.963	228.006
GATATATATG	366	198.065	371	199.689	1.85789	229.813	224.739
CTATATATAA	503	323.249	518	326.656	1.58576	238.832	222.411
CTATATATAC	376	210.02	385	211.789	1.81785	230.096	218.977
GTATATATAA	523	348.133	540	351.966	1.53424	231.139	212.859
GTATATATAG	356	198.179	358	199.804	1.79175	208.784	208.53
GATGAAGAAG	838	659.672	876	670.829	1.30585	233.763	200.512
CATATATATT	653	482.983	671	489.529	1.37071	211.583	196.942
AATATATATC	442	284.911	450	287.71	1.56407	201.283	194.096
GATATATATT	446	293.977	452	296.915	1.52232	189.948	185.901
TCAAAAAAAT	563	408.778	572	413.746	1.38249	185.264	180.22
AATATATATG	646	489.545	666	496.241	1.34209	195.956	179.15
TTTTTGTGTG	634	478.869	646	485.322	1.33108	184.748	177.915

Table A.96: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTTA	885	539.726	921	547.897	1.68097	478.342	437.655
TAAAAAAAAAT	853	551.834	883	560.315	1.5759	401.612	371.492
TTATATATATT	456	240.101	464	242.38	1.91435	301.312	292.494
AATATATATAA	444	229.845	450	231.982	1.93981	298.165	292.339
GTATATATATT	301	155.48	305	156.71	1.94627	203.104	198.839
CATATATATAC	230	99.9139	230	100.601	2.28626	190.191	191.767
TTATATATATG	301	159.352	305	160.624	1.89884	195.58	191.434
CATATATATAA	282	148.187	285	149.339	1.90841	184.187	181.448
AATATATATAC	282	155.047	283	156.273	1.81094	168.058	168.686
GTATATATATG	220	103.136	222	103.851	2.13767	168.658	166.668
TTTTTTTTTAA	1841	1711.49	1951	1776.64	1.09814	182.652	134.288
AATATATATAG	222	125.872	223	126.798	1.7587	125.9	125.966
TTATATATATC	172	82.9636	172	83.5081	2.05968	124.279	125.404
GATTTTTTTTT	1260	1141.57	1313	1172.1	1.12021	149.046	124.367
TTTTTTGTGTG	362	261.514	367	264.101	1.38962	120.754	117.706
AAAAAAAAATC	1238	1126.11	1288	1155.89	1.11429	139.389	117.268
GATATATATAA	167	83.7967	169	84.348	2.00361	117.446	115.163
TTTTTTTCTTT	2028	1916.36	2156	1997.26	1.07948	164.885	114.828
TATTTTTTTTG	482	380.571	487	385.187	1.26432	114.219	113.883
ATTTTTTTTTG	781	681.006	806	693.149	1.16281	121.576	107
GTTTTTTTTTC	530	433.907	537	439.606	1.22155	107.464	106.025
ATTTTTTTCTA	259	172.349	260	173.766	1.49626	104.772	105.493
CTTTTTTTTTC	513	420.122	521	425.531	1.22435	105.457	102.462
CTATATATATG	162	86.448	163	87.0209	1.87311	102.299	101.745
CATATATATAG	155	81.0974	156	81.6268	1.91114	101.041	100.405

Table A.97: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTC	452	256.306	457	258.947	1.76484	259.604	256.424
GAAAAAAAAAAC	410	229.732	418	231.984	1.80184	246.123	237.49
AATATATATATT	264	118.935	269	119.855	2.24438	217.468	210.507
TTATATATATAA	210	80.5132	212	81.0787	2.61475	203.767	201.324
ATTTTTTTTTTA	421	265.945	427	268.733	1.58894	197.73	193.384
TAAAAAAAAAAT	418	272.814	424	275.709	1.53786	182.485	178.357
ATTTTTTTTTTC	393	252.407	399	254.989	1.56478	178.649	174.008
CTTTTTTTTTTC	378	245.473	384	247.952	1.54869	167.964	163.185
GAAAAAAAAAAT	366	235.593	373	237.928	1.5677	167.704	161.234
TTATATATATAG	125	45.0672	125	45.3541	2.75609	126.727	127.52
CTATATATATAA	124	45.884	125	46.1768	2.70699	124.48	123.276
TTAAAAAAAAAA	1180	1071.27	1209	1098.99	1.1001	115.337	114.075
TTATATATATAC	127	53.7506	127	54.1014	2.34744	108.372	109.199
TTTTATTTTTTT	803	709.384	833	722.781	1.15249	118.225	99.5383
AAAAAGAAGAAG	477	388.27	488	393.233	1.24099	105.365	98.1734
TTTTTATTTTTT	785	694.239	837	707.149	1.18363	141.103	96.4506
GAAAAAAAAAAG	300	217.667	303	219.752	1.37883	97.3335	96.2447
GTATATATATAC	93	34.1956	93	34.4064	2.70298	92.4752	93.0467
CTTCTTCTTTTT	456	372.124	468	376.767	1.24215	101.482	92.6897
TAAAAAAAAAAC	346	266.032	351	268.821	1.3057	93.626	90.9368
CATATATATATT	138	71.6338	138	72.1251	1.91334	89.5416	90.4848
AAAAAAATAAAA	762	680.201	793	692.667	1.14485	107.272	86.5313
GTATATATATAA	110	51.2309	110	51.5629	2.13332	83.3447	84.0551
CTATATATATAC	83	30.6257	83	30.8124	2.69372	82.2466	82.7512
TTTTTTTTTTAA	1180	1102.13	1221	1131.33	1.07926	93.1379	80.554

Table A.98: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAATAAAATAAAT	150	73.3813	150	73.9241	2.02911	106.139	107.245
ATTTTTTTTTTTA	214	133.098	215	134.231	1.60172	101.282	101.626
GTTTTTTTTTTGT	275	193.743	275	195.61	1.40586	93.678	96.3162
TAAAAAAAAAAAT	211	134.145	213	135.289	1.57441	96.6769	95.5703
ATTTATTTTATTA	129	64.7229	132	65.1913	2.02481	93.1229	88.971
AAAGAAGAAGAAA	175	108.049	178	108.918	1.63425	87.4309	84.3845
TTTCTTCTTCTTT	182	119.396	182	120.381	1.51186	75.2285	76.7243
TTTTTTTTTTTGA	466	396.24	474	401.567	1.18038	78.605	75.5686
GAAAAAAAAAAAC	157	98.812	157	99.5897	1.57647	71.4644	72.6952
CTATATATACATT	88	39.6117	90	39.8799	2.25677	73.2543	70.2426
TCAAAAAAAAAAA	436	372.226	448	377.061	1.18814	77.2288	68.9492
AGTTTTTTTTTTT	698	635.823	712	647.262	1.10002	67.8724	65.1224
CAAATAAAATAAT	73	30.2223	74	30.4217	2.43248	65.7794	64.3772
ACAAAAAAAAAAC	207	153.611	208	154.976	1.34214	61.2072	61.7475
AAAAAAATAAAAA	493	435.747	515	441.929	1.16534	78.8035	60.8597
TTTTTTTTTTTCA	324	272.441	327	275.468	1.18707	56.0764	56.1564
GTTTTTTTTTTTA	190	142.203	190	143.437	1.32462	53.4142	55.0559
TTATATATATATT	86	45.5804	86	45.8941	1.87388	54.009	54.5987
ATTATTTTATTTG	74	35.9114	76	36.1521	2.10223	56.4679	53.5028
TTTTTATTTTTTT	501	450.38	521	456.894	1.14031	68.406	53.3641
AAAAAAAAAAACT	661	610.633	672	621.326	1.08156	52.6868	52.3889
TTTTTTATTTTTT	406	357.224	433	361.763	1.19692	77.831	51.964
AATATATATATAC	65	29.7661	65	29.9622	2.1694	50.3393	50.7661
AATGTATATATAG	69	33.8101	70	34.0353	2.05669	50.4767	49.221
ACAAAAAAAAAAT	214	170.826	216	172.4	1.2529	48.6999	48.2205

Table A.99: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTTTTA	122	78.861	122	79.4923	1.53474	52.26	53.2329
TATAAAATAAAAAT	89	52.8892	90	53.2872	1.68896	47.1703	46.3188
TAAAAAAAAAAAAT	112	75.204	112	75.801	1.47755	43.7234	44.609
TTTTTTTTCTTTTT	411	370.276	417	375.261	1.11123	43.979	42.8856
TAAAATAAAAATAT	90	56.135	91	56.5607	1.60889	43.2745	42.4845
GAATTTTTAAAAAC	37	11.766	37	11.8456	3.12353	42.1417	42.391
TTTTTATTTTTTTT	270	230.791	279	233.293	1.19592	49.9178	42.3656
GTTTTTAAAAATTC	40	14.6705	40	14.7705	2.7081	39.8499	40.1216
AGTTTTTTTTTTTT	509	470.966	516	478.203	1.07904	39.253	39.5299
TTTTTTTTTTTCTG	97	65.2491	97	65.755	1.47517	37.7112	38.4604
ATTTTTATTTTATA	76	46.1804	77	46.5221	1.65513	38.7987	37.8616
TAAAAAAAAAAAAC	109	77.2818	109	77.8983	1.39926	36.6179	37.4839
TTTTTTTTTTTTGG	489	453.577	493	460.397	1.07081	33.7306	36.7712
CATTTATTTTATTA	69	40.5511	69	40.8469	1.68923	36.175	36.6765
GTTATTAATTATTG	38	15.1674	38	15.2709	2.48839	34.6421	34.9006
CATTAATAAAAAAG	38	15.3151	39	15.4197	2.52923	36.1887	34.5323
CAGAAAAAAAAAAA	92	63.8678	92	64.3613	1.42943	32.8693	33.5776
CTATAAAATAAAAA	58	32.7108	58	32.9446	1.76053	32.8056	33.2188
GAAAATATTAAATC	33	12.1309	33	12.213	2.70203	32.8021	33.0247
AAAAAAAATAAAAA	259	228.087	265	230.549	1.14943	36.9061	32.9187
GTTTTTTTTTTTGT	128	99.0853	129	99.9159	1.29109	32.9574	32.7743
GAAAAAAAAAAAAT	80	53.3378	80	53.7395	1.48866	31.8302	32.4305
ATATTTTTATTTTA	70	44.1284	72	44.4532	1.61968	34.7205	32.2975
CAATAATTAATAAC	38	16.2587	38	16.3699	2.32133	32.0013	32.2605
CATTTTAAATGTAA	39	17.054	39	17.1709	2.27128	31.9934	32.26

Table A.100: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTATTTTTTTT	196	162.232	201	163.866	1.22661	41.0559	37.0607
AGTTTTTTTTTTTTT	406	372.102	409	377.314	1.08398	32.9807	35.3972
AAAAATAATATCATC	47	23.6361	47	23.8131	1.9737	31.9559	32.3065
ATTTTTTTTTTTTTA	79	52.9732	79	53.3987	1.47944	30.9413	31.5733
TAAAATAATATCATA	45	22.7088	46	22.8785	2.01062	32.1284	30.7759
TGTTTTTTTTTTTTG	80	54.4671	80	54.9061	1.45703	30.1121	30.7544
GATGATATTATTTTT	37	16.2803	37	16.4	2.2561	30.1046	30.3756
GTTTTTTTTTTTGTT	78	52.9958	79	53.4215	1.47881	30.9075	30.1467
AACATTATTAATTTA	28	9.9849	28	10.0571	2.78409	28.6698	28.8717
TATGATATTATTTTA	46	26.0772	47	26.2737	1.78886	27.3342	26.1086
GACAAAAAAAAAAAA	171	146.906	172	148.342	1.15948	25.451	25.9704
AAAAAAAATAAAAAA	169	145.743	173	147.165	1.17555	27.9805	25.0215
GAAATATTTATATTC	25	9.27813	25	9.34513	2.67519	24.6005	24.7804
AAAAAAAAAAAAACT	346	322.117	348	326.325	1.06642	22.3794	24.7478
TTTTTTTTTTTTGTC	167	144.054	167	145.455	1.14812	23.067	24.6837
TAAAAAAAAAAAAAT	69	48.3033	69	48.6871	1.41721	24.0597	24.6059
CAAAAACAAACAAAG	29	12.8151	29	12.9085	2.24659	23.473	23.6836
CAATATTTTAAAACA	24	9.1646	24	9.23077	2.6	22.9323	23.1049
TAAATTAATAATGTT	30	13.993	30	14.0952	2.12838	22.6608	22.8793
TAAAAAAATAAAAAT	54	35.3663	54	35.6388	1.5152	22.4396	22.8541
TTTTTTTTTCTTTTT	298	276.183	301	279.553	1.07672	22.2493	22.6566
AATAAAATTTTAAAG	30	14.1992	31	14.303	2.16737	23.979	22.4404
TAATATTTTAAAATT	38	21.1156	39	21.2727	1.83333	23.6393	22.3278
GATTGATTCAAATAA	24	9.49025	24	9.55882	2.51077	22.0941	22.2669
ATATTATTTTATTTG	47	29.3202	48	29.5429	1.62476	23.2972	22.1779

Table A.101: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTTGTTT	59	35.4896	60	35.7812	1.67686	31.0154	29.9896
ATATTTTATAGTTTAA	37	17.3104	38	17.4468	2.17805	29.5803	28.1055
AATATATATATATATA	509	481.984	525	489.984	1.07146	36.2382	27.7592
TTTTTTTTTTTTTTTG	738	711.229	745	726.147	1.02596	19.0953	27.2681
TTAAACTATAAAATAT	25	8.76727	25	8.83495	2.82967	26.004	26.1963
GGAAGAAGAAGAAGAG	34	15.9861	34	16.1117	2.11027	25.3917	25.6577
TTATTTTATAGTTTAC	43	23.683	43	23.8723	1.80125	25.3046	25.6471
GATATTATTTTATTTG	44	25.3286	45	25.5319	1.7625	25.503	24.2992
GTAAACTATAAAATAA	62	43.2116	62	43.5728	1.42291	21.8675	22.3836
CAAATAAAATAATATC	48	31.4224	49	31.6782	1.54681	21.3734	20.3367
ATATTTTTATTTTATA	42	26.0019	43	26.2109	1.64054	21.286	20.139
GAAAAAAAAAAAAAAA	277	258.221	279	261.416	1.06726	18.1627	19.4462
TCATATCATAAATAAA	19	7.07238	19	7.12676	2.66601	18.6311	18.7766
TTTTAATTAATTAAAA	11	2.00281	11	2.01802	5.45089	18.6536	18.7368
TATATATATATATATT	486	468.036	498	475.681	1.04692	22.8345	18.3039
TTCATCATCATCATCT	16	5.09809	16	5.1371	3.1146	18.1776	18.2996
AAAAAAAATAAAAAAA	118	101.271	121	102.227	1.18364	20.3996	18.04
TTTTTTTTTTTTTGTC	138	121.294	138	122.485	1.12667	16.459	17.8065
CTTCTTTTTTTTTTTT	78	62.2271	78	62.7692	1.24265	16.945	17.6217
GTTTTTTTTTTTTTTT	513	495.989	518	504.353	1.02706	13.8296	17.2994
AAACAAAAAAAAAAAC	37	23.2518	37	23.4375	1.57867	16.8935	17.1879
AAAAAAAAATAAAAAA	99	83.455	102	84.2151	1.21118	19.5431	16.9104
TATAAAATAAAAATAT	53	38.5698	54	38.8889	1.38857	17.7269	16.8446
TTTTATTTTATTAATT	10	1.88155	10	1.89583	5.27473	16.6293	16.7049
AACAAAAAAAAAAAAC	29	16.454	30	16.5833	1.80905	17.784	16.4352

Table A.102: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATAATTTAATTAATTAT	16	4.03756	16	4.07042	3.9308	21.9015	22.0312
TTTTTTGGCGGGAAAAA	16	4.22485	17	4.25926	3.9913	23.53	21.3057
TTATATATAAAAGTTTT	27	12.5831	27	12.6875	2.12808	20.3909	20.6141
TTTTTCCCGCCAAAAAA	15	3.80325	15	3.8342	3.91216	20.4614	20.5829
AAAATTAACAATCAAAA	22	8.80474	22	8.87719	2.47826	19.9663	20.1465
ATAATTAATTAAATTAT	12	2.42104	12	2.44068	4.91667	19.1116	19.2085
TTTTGATTGTTAATTTT	20	8.09675	20	8.16327	2.45	17.9218	18.0854
TTTTTTTTTTTTTTTG	571	553.6	577	563.827	1.02336	13.3257	17.6703
TAAATTAACAATCAAAC	30	16.7022	30	16.8421	1.78125	17.3195	17.5697
AAAACTTTTATATATAA	22	9.98541	22	10.0678	2.18519	17.1974	17.3782
AAAAAAAAAAAAAAAAC	343	326.615	347	331.244	1.04757	16.1254	16.7892
GTTTGATTGTTAATTTA	25	13.156	25	13.2653	1.88462	15.8431	16.05
TAAAAAAAGAAGAAGAT	23	11.7424	23	11.8397	1.94262	15.2728	15.4626
TTTTTTATTTTATTTTT	61	47.4548	61	47.8794	1.27403	14.7735	15.3168
AAAAAAATTAATATGAC	25	13.5966	25	13.7097	1.82353	15.0193	15.2264
TAAAAAATTAATATGAT	27	15.6755	27	15.8065	1.70816	14.4563	14.681
AAATATTTATAAAATTA	14	5.06506	14	5.10638	2.74167	14.1199	14.2337
TTTTTTTTTTCTTTTTT	132	119.469	133	120.698	1.10193	12.909	13.1667
GTATATATAAAAGTTTA	13	4.88206	13	4.92187	2.64127	12.6264	12.732
CTATATATATATTTTTG	8	1.65328	8	1.66667	4.8	12.5489	12.6134
AATATATATATATATAT	486	473.691	502	481.721	1.0421	20.6998	12.4679
TTATTAATTTATCGATT	16	7.62969	16	7.69231	2.08	11.7179	11.8487
TTTTTTTTTTTCTTTTT	90	79.1228	91	79.8773	1.13925	11.8635	11.5927
GTTTTTTTTTTGTGAAC	13	5.36528	13	5.40909	2.40336	11.3993	11.505
TAAAAAAAAAAAAAAAT	36	26.1982	36	26.4223	1.36249	11.1352	11.4418

Table A.103: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GATCAACATTAATAAAAT	35	14.8127	35	14.9438	2.34211	29.7868	30.0951
AATCAACATTAATAAAAA	51	30.6635	51	30.9438	1.64815	25.4823	25.9464
ATTTTATTAATGTTGATC	26	10.7712	26	10.8657	2.39286	22.6847	22.9118
TTTTTATTAATGTTGATT	39	23.6525	39	23.8657	1.63415	19.1537	19.5036
TAAAAAAAAAAAAAAAAT	31	19.2251	31	19.3968	1.5982	14.5352	14.8108
TTTGTAAATCTTTTAAAG	13	4.17753	15	4.21368	3.55984	19.0457	14.758
TTATATATATATATATAA	24	13.0297	25	13.1445	1.90193	16.0717	14.6598
TGAACTTATTTACAATGT	20	9.91318	20	10	2	13.8629	14.0373
TAGAAGAAGAAGAAGAAC	6	0.733787	6	0.740088	8.10714	12.5565	12.6078
ATTTTTGTTTTTTTTTTG	14	5.82287	14	5.87342	2.38362	12.1607	12.2817
TTTATAATAATAATTAAT	9	2.54945	9	2.57143	3.5	11.2749	11.3521
GGAACTTATTTACAATGG	16	7.93084	16	8	2	11.0904	11.2293
GGTAAACTATAAAATAAT	5	0.559712	5	0.564516	8.85714	10.9061	10.9489
AAAAAAAAAAAAAAAAAG	282	271.403	285	275.106	1.03596	10.0699	10.801
GATCATCATCATCATCAA	6	1.00795	6	1.01661	5.90196	10.6517	10.703
TATATATATATATATATT	417	406.832	425	413.422	1.02801	11.7388	10.2941
TGAAGAAGAAGAAGAAGC	11	4.31849	11	4.35586	2.52533	10.1901	10.2849
CCATTGTAAATAAGTTCC	9	3.1982	9	3.22581	2.79	9.23437	9.31174
TAACATTAATAAAATAAT	6	1.30131	6	1.3125	4.57143	9.11895	9.17032
AAAAAGATTTACAAGAGT	6	1.30875	6	1.32	4.54545	9.08477	9.13613
AAGAGAGAGAGAGAGAGG	9	3.27878	9	3.30709	2.72143	9.01041	9.08779
GTTATAATAATAATTAAA	21	13.6792	21	13.8	1.52174	8.81693	9.0015
ATTTTTTTTTTCTTTTCA	8	2.77606	8	2.8	2.85714	8.39858	8.46728
CAGAGAGAGAGAGAGAGC	4	0.491842	4	0.496063	8.06349	8.34939	8.38357
AATCGATAAATTAATAAG	18	11.4149	18	11.5152	1.56316	8.04075	8.19819

Table A.104: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATCTCTCTCTCTCTCTCTT	15	4.45186	15	4.49267	3.33878	18.0841	18.2209
GAAAAAAAAAAAAAAAAC	26	14.3314	26	14.4654	1.79739	15.2447	15.4867
GTTATTAATTATTATAGTA	8	1.51808	8	1.53191	5.22222	13.2234	13.2959
ATCTTCTTCTTCTTCTTCC	15	6.4057	15	6.46465	2.32031	12.6255	12.7629
CTTTTTTTTTTTTTTTTTC	25	15.294	25	15.4373	1.61946	12.0523	12.2854
CGAGAGAGAGAGAGAGAGC	4	0.218702	4	0.22069	18.125	11.5892	11.6254
CAAAAAAAAAAAAAAAAAA	301	289.625	309	293.824	1.05165	15.5618	11.5958
TAATATATATATATATATT	17	8.63335	17	8.71315	1.95107	11.3625	11.5189
AAATATATATATATATATA	208	196.934	209	199.446	1.0479	9.77904	11.3711
AAAAAAAAAAAAAAAAAAG	223	211.954	225	214.717	1.04789	10.5252	11.3292
TTTAATAAAATTAATATTT	13	5.63572	13	5.6875	2.28571	10.7468	10.8657
TACTATAATAATTAATAAC	7	1.57654	7	1.59091	4.4	10.3712	10.4347
ACTTTTTTTTTTTTTTTTT	99	89.3322	99	90.2921	1.09644	9.11494	10.173
GTTTTTTTTTTTTTTTGTT	12	5.29822	12	5.34686	2.24431	9.70076	9.81043
TTTTTAATTAATTTAATTA	6	1.30066	6	1.3125	4.57143	9.11895	9.17335
TAAAAAAAAAAAAAAAAAT	21	13.6547	21	13.7822	1.52371	8.8441	9.03925
ATTTTTTTTTTTTTTTTTA	21	13.7522	21	13.8806	1.5129	8.69465	8.88984
ATTTTTTTTATTTTTTTTG	11	4.95454	11	5	2.2	8.67303	8.77349
CTATTAATTTATAGAGGTA	4	0.45045	4	0.454545	8.8	8.69901	8.73521
GCCTTGTAAATCATTCTAC	6	1.427	6	1.44	4.16667	8.5627	8.61711
TATTTTTTTTTTTTTTTTG	13	6.73476	13	6.79677	1.91267	8.43052	8.54967
CAAAAAAAAAAAAAAAAGT	21	13.9817	21	14.1123	1.48806	8.34695	8.54222
AAGTTTTTTTTTTTTTTT	67	58.9829	67	59.5833	1.12448	7.86022	8.53877
CTTTTTAAAGTTAGTATAA	8	2.75742	8	2.78261	2.875	8.44842	8.52116
ATTTTTTTTTTTTTTTTTG	35	27.5622	35	27.8267	1.25779	8.02738	8.36159

Table A.105: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAGATTTGATATCTAAATTT	29	13.6749	29	13.8095	2.1	21.5162	21.8003
TAGATTTGATATCTAAATTA	31	17.6346	33	17.8095	1.85294	20.3535	17.4879
AAATTTAGATATCAAATCTT	21	9.45482	22	9.54717	2.30435	18.3655	16.758
TATATATATATATATATATT	355	338.776	359	344.175	1.04307	15.1393	16.6059
ATTAATTAAATTTAATTAAA	23	11.6207	23	11.7347	1.96	15.4777	15.7022
TTTAATTAAATTTAATTAAT	24	12.6108	24	12.7347	1.88462	15.2094	15.4441
AAATTTAATTAATTAAATTA	30	18.2375	33	18.4186	1.79167	19.2438	14.9315
TAATTTAGATATCAAATCTA	29	17.3749	30	17.5472	1.70968	16.0891	14.8558
TAATTGTTATTAAGTAACGC	18	8.751	18	8.83636	2.03704	12.8069	12.9817
AATATATATATATATATATA	375	362.63	383	368.573	1.03914	14.706	12.5785
CAATTGTTATTAAGTAACGT	25	15.3034	25	15.4545	1.61765	12.0243	12.27
GTTTTTTTTTTTTTTTTTC	19	10.0492	19	10.1474	1.87239	11.9171	12.102
GACAAAAAAAAAAAAAAAA	62	51.3174	62	51.8586	1.19556	11.0741	11.7245
TAATTTAATTAATTAAATTT	33	23.6925	38	23.9302	1.58795	17.5729	10.9345
CTAAAATCCCTTGTAAATCG	4	0.285719	4	0.288462	13.8667	10.518	10.5562
TTTTTTTTTTTTTTTTTTG	252	242.134	253	245.552	1.03033	7.56024	10.0647
GCGTTACTTAATAACAATTA	8	2.43804	8	2.46154	3.25	9.42924	9.50597
GTTTTTTTTTTTTTTTTTA	21	13.4255	21	13.5577	1.54894	9.18899	9.39465
CATTTTCCCGCCAAAAACGA	6	1.29265	10	1.30508	7.66234	20.3632	9.21037
TTTTTTTTTTTTTTTTTGT	120	111.29	120	112.588	1.06583	7.65054	9.0422
ACGTTACTTAATAACAATTG	33	25.1318	33	25.3846	1.3	8.65802	8.98831
ACCCGCGGTATACCGCGGGA	16	9.3778	16	9.46939	1.68966	8.39239	8.54789
TATATATATATATATATATC	121	112.825	122	114.144	1.06882	8.12006	8.46429
AAATCATTTTGGTCCTTTTT	10	4.30618	10	4.34783	2.3	8.32909	8.42535
TTTTTTTTTTTTTTTTTAA	97	88.957	98	89.9575	1.0894	8.3917	8.39614

A.6 The *Arabidopsis thaliana* Coding Sequences

A.6.1 Timing Information

Table A.106: Full Size and Timing Information, Coding Sequences

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	3.869915	4.582698	57.985799	8.452613
2	0	0.000001	6.992290	3.276406	102.946337	10.268696
3	1	0.000003	10.998605	5.230845	56.569034	16.22945
4	2	0.000012	16.416183	4.194964	16.042124	20.611147
5	3	0.000046	22.291380	5.509885	5.223922	27.801265
6	4	0.000186	28.603787	4.760546	2.872546	33.364333
7	5	0.000743	35.775706	11.545457	2.177797	47.321163
8	6	0.002971	47.330480	34.618181	2.214564	81.948661
9	7	0.011883	61.626166	130.087176	3.532948	191.713342
10	8	0.047402	78.750148	454.900436	9.520215	533.650584
11	9	0.182296	97.666313	1567.218623	32.183024	1664.884936
12	10	0.600019	119.944626	4419.175887	93.437479	4539.120513
13	11	1.519743	143.326199	5879.959238	184.019170	6023.285437
14	12	2.959445	165.340177	8520.296843	266.080077	8685.63702
15	13	4.726223	187.543875	9681.475481	327.213557	9869.019356
16	14	6.639012	210.104767	11467.195191	376.015126	11677.299958
17	15	8.607813	231.880176	10620.055543	440.857094	10851.935719
18	16	10.599029	257.147598	11482.797700	462.840768	11739.945298
19	17	12.600701	284.221933	9657.351031	505.420799	9941.572964
20	18	14.608070	304.120663	9416.630150	560.815682	9720.750813

A.6.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.107: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
C	27133	27133	6685164	6.68516e+06	1	0	1.8392e-05
G	27133	27133	7844064	7.84406e+06	1	0	1.57722e-06
T	27133	27133	8941769	8.94177e+06	1	0	1.45317e-07
A	27133	27133	9443813	9.44381e+06	1	0	4.75954e-08

Table A.108: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CC	27100	27127.4	1287607	1.35779e+06	0.948311	-68337.5	-27.423
AC	27123	27132	1688758	1.91809e+06	0.880439	-215037	-8.98662
CA	27131	27132	2140560	1.91809e+06	1.11599	234904	-0.988094
CG	27099	27130.4	1124211	1.59317e+06	0.705645	-391949	-31.3366
AT	27133	27132.8	2347194	2.56554e+06	0.914891	-208783	0.177378
AG	27131	27132.6	2445608	2.25059e+06	1.08665	203229	-1.5965
AA	27132	27132.9	2940690	2.70959e+06	1.08529	240688	-0.876505
GC	27125	27130.4	1460011	1.59317e+06	0.916419	-127431	-5.35418
TA	27132	27132.8	1556664	2.56554e+06	0.606758	-777749	-0.82261
GG	27128	27131.8	1795815	1.86935e+06	0.960662	-72070.4	-3.83591
GA	27132	27132.6	2778766	2.25059e+06	1.23468	585799	-0.596544
CT	27122	27131.6	2132785	1.81612e+06	1.17436	342797	-9.64011
TC	27132	27131.6	2248786	1.81612e+06	1.23824	480541	0.358175
GT	27132	27132.4	1803897	2.13095e+06	0.846523	-300563	-0.442057
TG	27133	27132.4	2478428	2.13095e+06	1.16306	374383	0.557946
TT	27130	27132.7	2657889	2.42916e+06	1.09416	239178	-2.74801

Table A.109: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAC	26702	26541.1	370129	278365	1.32965	105456	161.398
ATG	27133	27054.2	737497	650582	1.1336	92478.5	78.9461
CGT	26527	26449.4	277603	258534	1.07376	19755.1	77.7296
GGT	26914	26878.6	484879	412983	1.17409	77819.6	35.4239
TGG	27050	27017.8	700233	567410	1.23409	147281	32.2322
GTT	27030	26999.2	644324	536198	1.20165	118361	30.8369
CCG	26206	26182.4	266089	216531	1.22888	54840.8	23.6264
CAA	27041	27059.5	764926	666545	1.1476	105309	-18.492
GCT	26921	26942	545920	465791	1.17203	86656.8	-21.0035
TGA	27080	27102	757981	877985	0.863319	-111401	-21.9565
TTG	27053	27078.5	770204	736698	1.04548	34256.3	-25.4542
GTA	26643	26669.3	307589	314039	0.979462	-6383.01	-26.2431
CCA	26851	26877.6	451396	412286	1.09486	40908.9	-26.5876
ACC	26660	26702	367174	325266	1.12884	44498.7	-41.9977
TTA	26893	26938.9	456170	462709	0.985867	-6492.92	-45.8501
TTC	27013	27060.1	714661	668439	1.06915	47784.9	-47.0613
ACG	26504	26564.1	294279	283990	1.03623	10473.1	-60.0575
TAT	26777	26837.6	468358	386898	1.21055	89490	-60.5655
TAA	26894	26959.9	404110	484726	0.833687	-73506.7	-65.8251
AAG	27017	27083.8	914115	761533	1.20036	166938	-66.6929
ATT	27001	27068.7	602248	697690	0.863203	-88593.9	-67.6493
TCA	27006	27074.5	688276	720051	0.955871	-31063.7	-68.4509
TGT	26940	27019.2	558971	569963	0.980714	-10885.6	-79.0516
TCG	26740	26822.1	354120	378167	0.936411	-23265.9	-81.9722
AAC	26907	26992.2	561760	525859	1.06827	37099.6	-85.101

Table A.110: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CGCC	18911	16891.3	51650	33922.3	1.5226	21714.5	2135.86
GGCG	20841	18951.7	62849	44149.2	1.42356	22195.7	1980.52
CGAC	21657	20245.1	67534	52726	1.28085	16716.2	1460.08
CGTC	22175	20856.3	75313	57647.2	1.30645	20132.1	1359.52
GTCG	22206	21009.2	69728	58988.6	1.18206	11662.6	1230.27
GACG	22895	21933.1	85709	68281.8	1.25522	19482.9	982.68
TCGT	23844	23291.3	95527	87443.3	1.09244	8446.29	559.182
CTAA	24182	23816.6	103808	97752.6	1.06195	6239.15	368.202
ATGG	26373	26108.3	222456	208366	1.06762	14556.2	266.037
ACGA	24045	23822.2	106737	97875.7	1.09054	9250.81	223.827
CGGT	22857	22635.2	88121	77129.9	1.1425	11739.5	222.93
CACC	22958	22790.5	95944	79364.4	1.2089	18202	168.118
CATC	25051	24890.5	148080	128745	1.15018	20718.9	161.061
GTTT	25878	25734.7	198911	173779	1.14462	26867.1	143.726
ATCG	23708	23592.4	94622	93080.9	1.01656	1553.84	115.874
GCTA	23850	23753.1	103827	96384.2	1.07722	7723.07	97.0881
TCGC	21085	21001.1	60987	58916.2	1.03515	2106.76	84.0838
CGAT	24004	23937.7	103131	100473	1.02645	2692.79	66.3551
TTAC	24301	24258.6	119787	108464	1.1044	11894.7	42.4065
GCGA	22096	22060.5	70435	69752.8	1.00978	685.496	35.5748
CCAC	22657	22624.1	89935	76975.4	1.16836	13994.1	32.9114
GCGG	20345	20319.5	61200	53290.4	1.14842	8469.5	25.5494
AAAC	25620	25603	188342	164519	1.1448	25470.1	16.9626
TAGC	21625	21609.8	71524	64772.8	1.10423	7091.45	15.1807
ACGC	19730	19716.8	50835	48960.2	1.03829	1910.2	13.1772

Table A.111: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CGGCG	10248	8541.89	19179	11363.5	1.68777	10038.4	1866.16
CGCCG	10420	8905.81	19890	12017.4	1.6551	10021.7	1636.18
CCTAA	13297	11764.1	22010	17917.6	1.2284	4527.73	1628.72
TAGGG	8641	7219.77	12065	9138.2	1.32028	3352.2	1552.74
CGTCG	11407	9956.8	19181	14018.8	1.36823	6013.61	1551.02
CGACG	11670	10329.3	20570	14771.9	1.39251	6810.9	1424.16
ATGGC	19066	17741.2	41006	37794.8	1.08496	3343.94	1373.08
TAGAG	16484	15259.8	35477	27774.6	1.27732	8683.39	1272.06
GTCAC	13933	12786.8	23385	20434.9	1.14437	3153.48	1196.11
CCCGA	9452	8353.36	13408	11032.1	1.21536	2615.08	1167.91
TAGTG	13157	12051.8	23178	18600.6	1.24609	5099.32	1154.34
CGAGT	13423	12408.1	22459	19473	1.15334	3204.05	1055.35
GTGAC	13690	12715	22805	20249.6	1.12619	2710.21	1011.5
AACCC	13163	12191.4	22121	18938.9	1.16802	3435.62	1009.28
TAGCG	7841	6901.58	10170	8635	1.17777	1664	1000.64
CGAAT	13100	12190.6	21469	18937	1.13371	2694.25	942.453
ACTCG	12981	12079.4	20898	18667.1	1.11951	2359.15	934.401
TTAGG	11913	11014.3	18834	16222.6	1.16097	2811.09	934.379
TAAGG	13548	12730.7	23912	20290.2	1.1785	3927.37	842.956
CCCTA	8692	7927.81	11827	10301.8	1.14805	1632.87	799.888
GATCC	17199	16423.5	34932	32051.8	1.08986	3005.94	793.496
TAACG	10316	9560.26	15061	13243.1	1.13727	1937.37	784.854
CATCG	14729	13973.1	25357	23704.5	1.06971	1708.82	776.008
GGGTT	16225	15470.2	33880	28501.6	1.18871	5856.67	772.948
TCGTC	15440	14696.1	29080	25916.2	1.12208	3349.46	762.385

Table A.112: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATGGCG	7238	6248.58	8680	7645.37	1.13533	1101.68	1063.92
GAGTAT	7629	6646.43	9966	8247.77	1.20833	1885.94	1051.86
GGGTTT	9519	8562.14	13735	11410.8	1.20368	2546.23	1008.44
TAATAA	5043	4135.41	6423	4710.75	1.36348	1991.38	1000.6
TAGTAG	4030	3178.31	5042	3511.2	1.43598	1824.43	956.789
TAGTAA	4208	3365.28	4973	3739.79	1.32975	1417.28	940.389
GATCTT	11863	11004	18513	16217.4	1.14155	2450.89	891.702
GGAAGT	9856	9009.68	14401	12220	1.17848	2365.02	884.883
TCCGGC	4692	3887.08	5814	4392.37	1.32366	1630.25	883.042
ACCCTA	4029	3254.85	4662	3604.45	1.2934	1199.42	859.678
GATGGT	11263	10455.7	17457	15048.9	1.16002	2591.28	837.665
CATGCC	4098	3366.45	4795	3741.23	1.28166	1189.92	805.83
ACGAAG	8623	7878.34	11463	10228.5	1.12069	1306.12	778.794
TAAACC	6227	5535.91	7772	6607.87	1.17617	1261.14	732.539
CAGGAC	4311	3654.01	5006	4098.18	1.22152	1001.67	712.8
GCTCTT	10226	9548.11	14904	13233.3	1.12625	1771.98	701.399
TAGACC	3453	2844.82	3942	3110.09	1.26749	934.396	668.995
GAGTTT	12303	11654.1	19940	17680.9	1.12777	2397.69	666.602
GAGCTT	12200	11568.9	19631	17483.9	1.1228	2273.85	648.017
CATGAC	5078	4470.91	5975	5149.18	1.16038	888.764	646.558
CGAACC	3578	2986.53	4003	3279.51	1.22061	798.002	646.514
GTCTTT	7932	7311.27	10500	9294.13	1.12975	1280.92	646.365
TAATGA	7869	7248.82	10779	9193.64	1.17244	1714.81	645.988
CAAGAC	9214	8590.44	12631	11461.2	1.10207	1227.62	645.667
TTTCCG	6182	5576.7	7586	6665.87	1.13804	980.897	637.02

Table A.113: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAAGAAC	5767	4445.98	7025	5120.91	1.37183	2220.9	1500.3
GTTCTTG	6146	4952.52	7684	5801.18	1.32456	2159.81	1326.95
GGAAGAG	7077	6165.57	9341	7528.96	1.24068	2014.45	975.699
CATCTTC	5138	4258.22	6220	4874.52	1.27602	1516.11	964.985
CAAGATC	4652	3814.43	5399	4304.06	1.2544	1223.72	923.449
GGATTTG	4986	4146.64	5891	4729.53	1.24558	1293.66	919.103
CGTCTTC	3296	2503.08	3712	2710	1.36974	1167.88	907.02
GGAGTTG	4109	3335.42	4693	3706.38	1.26619	1107.62	857.066
TAGGGTT	2238	1532.4	2483	1610.82	1.54145	1074.46	847.636
GAACAAG	5275	4557.28	6426	5268.43	1.21972	1276.34	771.483
GAAGACG	4219	3514.05	4900	3927.12	1.24773	1084.51	771.36
GGAAGTG	3812	3115.78	4384	3438.39	1.27502	1065.13	768.778
CAAAATC	4313	3615.06	4962	4053.07	1.22426	1003.98	761.353
AACCCTA	2509	1858.56	2823	1972.95	1.43085	1011.4	752.908
GGAGATG	6590	5882.78	8248	7113.05	1.15956	1221.04	748.128
GAGCAAG	4157	3494.45	4800	3902.79	1.22989	993.239	721.725
GCTCTTG	4191	3532.35	4883	3949.88	1.23624	1035.56	716.562
GTGATTG	3139	2502.86	3445	2709.75	1.27134	827.038	710.885
CAACTTC	3921	3286	4553	3645.76	1.24885	1011.77	692.743
GATCATG	3538	2911.25	3950	3192.13	1.23742	841.461	689.838
CGATTTC	2642	2035.48	2925	2172.39	1.34644	870.093	689.05
CTTCATC	5079	4438.82	6044	5111.47	1.18244	1012.85	684.273
GAAATCG	2969	2367.83	3249	2552.9	1.27267	783.385	671.746
GGAGCTG	3995	3377.8	4609	3758.53	1.22628	940.158	670.439
CAAATCC	3468	2861.7	3887	3132.94	1.24069	838.294	666.423

Table A.114: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ACTTCTTG	1317	1008.36	1379	1044.45	1.32032	383.184	351.682
TAAGATTG	956	684.288	983	702.056	1.40017	330.874	319.667
AAAGAAGG	2440	2148.91	2672	2303.39	1.16003	396.646	309.969
TGAGATTG	2052	1765.99	2232	1871.05	1.19291	393.719	308.015
CTCTTCTC	2352	2069.68	2598	2213.11	1.17391	416.572	300.752
CTCTGTTT	1865	1601.17	2055	1688.02	1.2174	404.256	284.467
GAGATTGA	2432	2171.36	2702	2329.05	1.16013	401.336	275.694
CAAGAAGC	2223	1964.61	2393	2094.05	1.14276	319.344	274.685
CAGAAGAT	2146	1889.64	2336	2009.56	1.16244	351.623	273.015
TCATCTTC	2591	2333.86	2854	2515.85	1.13441	359.923	270.813
GGAGATGA	2739	2488.44	2983	2695.29	1.10675	302.552	262.768
GGAAGAAC	1671	1432.58	1767	1502.68	1.1759	286.313	257.241
AAAGAAGT	2097	1855.56	2270	1971.28	1.15154	320.29	256.512
ATGGCTTC	1600	1363.27	1652	1427.02	1.15766	241.85	256.187
GAAGATGA	4828	4579.46	6139	5302.79	1.15769	898.922	255.165
GAGGATGA	2433	2194.94	2801	2356.04	1.18886	484.557	250.527
TCTTCATC	2587	2349.54	2830	2533.98	1.11682	312.678	249.07
TGCTTCTG	1760	1529.44	1894	1608.93	1.17718	308.948	247.129
GCTTCTTG	1580	1354.7	1665	1417.69	1.17445	267.728	243.072
GAAGGAGA	3728	3494.25	4354	3906.01	1.11469	472.748	241.404
TTGATGAG	2265	2036.4	2475	2175.31	1.13777	319.447	240.98
GAAGACGA	2295	2070.2	2562	2213.69	1.15734	374.376	236.591
TCAACAAG	1695	1478.4	1806	1552.86	1.16301	272.731	231.749
CAAGAAGT	1666	1451.55	1759	1523.44	1.15462	252.895	229.56
TGATTTTG	2015	1801.6	2201	1910.84	1.15185	311.153	225.564

Table A.115: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAGAAGAAA	1810	1615.97	1990	1705.85	1.16657	306.6	205.237
AAAGAAGAG	1381	1193.58	1484	1244.15	1.19278	261.609	201.423
TGAAGAAGT	1022	874.393	1056	902.878	1.16959	165.428	159.419
TGAAGAAGG	1204	1059.46	1257	1099.97	1.14276	167.739	153.977
ATCTTCTTG	631	496.352	641	506.877	1.26461	150.482	151.453
TGAAGAAGC	1399	1257.03	1477	1312.77	1.1251	174.098	149.7
CAAGAAGAT	1119	982.199	1165	1017.43	1.14505	157.792	145.914
AAGAGATTG	1050	919.249	1088	950.452	1.14472	147.053	139.638
CTGATGATT	548	426.109	562	434.256	1.29417	144.921	137.866
TTTCTTCTG	588	465.78	595	475.234	1.25202	133.729	137.012
TTGATGATC	609	491.104	628	501.442	1.25239	141.333	131.034
GAGAAGATT	957	839.138	987	865.576	1.14028	129.568	125.777
AGATTCTTG	815	700.49	838	719.624	1.1645	127.618	123.398
TGAAGAAAT	1055	939.046	1105	971.487	1.13743	142.293	122.835
CTTCTTCAT	1029	913.689	1064	944.548	1.12647	126.706	122.3
GAGAAGTAT	475	369.066	485	375.5	1.29161	124.107	119.861
GAGAAGCTT	1123	1009.59	1178	1046.65	1.1255	139.271	119.554
CTCTTCTTT	642	534.433	652	546.372	1.19333	115.237	117.732
AGAGAAGAG	1189	1078.3	1256	1120.15	1.12128	143.774	116.202
GACAACAAA	431	329.676	437	335.04	1.30432	116.104	115.507
GAGAAGTTT	746	640.272	765	656.603	1.16509	116.889	114.013
CTCTTCTTG	668	564.774	677	577.902	1.17148	107.146	112.133
AGAGATTTG	676	573.53	697	587.012	1.18737	119.702	111.123
ACTTCTTCA	556	456.437	569	465.575	1.22214	114.145	109.708
CAGAAGAAC	458	363.317	473	369.589	1.2798	116.691	106.069

Table A.116: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TGAAGAAGAT	869	694.208	900	713.64	1.26114	208.814	195.153
TTGATGATGT	311	197.685	315	200.301	1.57263	142.616	140.92
CTTCTTCCTC	710	588.539	739	603.145	1.22524	150.12	133.211
GATGAAGAAG	1236	1114.97	1315	1160.49	1.13314	164.37	127.378
GGAGAAGAAG	1126	1007.19	1193	1044.97	1.14166	158.049	125.554
AGGAAGAAGA	1337	1219.8	1464	1273.56	1.14953	204.017	122.66
TTGATGAAGC	361	257.049	371	260.9	1.422	130.617	122.599
TACAACAACT	149	67.7763	154	68.4162	2.25093	124.947	117.372
GTTCTTCTTG	193	105.259	194	106.368	1.82386	116.585	117.009
CTTCTTCATC	589	486.755	602	497.355	1.2104	114.954	112.302
AAAGAAGAAA	669	568.718	691	582.494	1.18628	118.037	108.645
AAGAAGATGA	1204	1106.22	1287	1151.09	1.11807	143.635	101.975
AAACAACAAA	201	121.45	204	122.786	1.66142	103.565	101.265
ATGAAGAAGA	1199	1103.82	1283	1148.5	1.11711	142.08	99.1728
CAGAAGAAGC	325	240.866	328	244.36	1.34228	96.5541	97.3645
AGGAGAAGAA	904	813.138	973	838.824	1.15996	144.376	95.7594
TAGAAGAAGC	220	143.306	223	144.975	1.53819	96.0259	94.3014
CATCTTCTTC	581	494.104	594	504.973	1.1763	96.4508	94.1243
TCTTCTTCCT	693	606.369	713	621.742	1.14678	97.6497	92.5436
TGGAGAAGAT	540	457.476	550	467.039	1.17763	89.9276	89.5567
ATGATGAAGA	901	815.777	969	841.612	1.15136	136.577	89.5276
AGATGATGAA	526	445.46	543	454.614	1.19442	96.4699	87.4175
GAAGAAGACG	577	496.156	593	507.099	1.1694	92.7969	87.0996
AAGGAGGAGA	358	280.912	368	285.317	1.28979	93.6497	86.8122
GAGGAAGAAG	1261	1177.58	1395	1227.94	1.13605	177.94	86.3051

Table A.117: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAAGAAGATGA	790	707.977	835	728.703	1.14587	113.698	86.601
GAGAAGAAGAG	334	277.608	341	282.169	1.20849	64.5768	61.7675
CTCTTCTTCTC	206	154.291	207	156.268	1.32465	58.1974	59.5407
AAAGAAGAAGC	205	156.035	208	158.042	1.3161	57.1326	55.95
CAGAAGAAGAG	184	136.535	184	138.213	1.33128	52.65	54.8976
AGGAAGAAGAA	693	641.314	723	658.8	1.09745	67.2312	53.7146
TTGATGATGAG	134	92.2622	134	93.2768	1.43658	48.544	50.0095
TTGATGATGAA	176	135.519	179	137.181	1.30485	47.6292	46.001
GAAGAAGACGA	393	350.027	406	356.528	1.13876	52.7562	45.5093
ATGGCTTCTTC	167	127.658	167	129.194	1.29263	42.8659	44.863
CTGATGATGAA	144	105.472	147	106.673	1.37804	47.1378	44.8365
GGATGGAACTC	64	31.9787	64	32.2742	1.98301	43.8154	44.4041
AGAGGAGGAGA	153	115.272	154	116.617	1.32056	42.8211	43.3212
GAAGAAGCAGA	311	270.727	315	275.121	1.14495	42.6393	43.1303
CAGAAGAAGAT	167	129.499	167	131.064	1.27419	40.4657	42.4717
TTGAAGAAGAG	215	177.964	217	180.367	1.2031	40.1245	40.6477
TAGAAGAAGAT	99	66.0476	99	66.7234	1.48374	39.0618	40.0697
CAAGAAGAGAT	119	85.6529	120	86.5783	1.38603	39.1731	39.1296
TTCCTCCTCCG	110	77.868	111	78.6916	1.41057	38.1833	38.0012
ATGTTCCAAGT	81	50.9335	82	51.4323	1.59433	38.2491	37.5781
AGAAGAGGAGA	195	160.978	198	163.071	1.21419	38.4277	37.3881
CGTGATGGATA	42	17.3205	42	17.4731	2.40369	36.8342	37.2028
AAAGAAGAAGG	185	151.328	189	153.253	1.23325	39.6251	37.1681
TGTTGGAGATG	155	122.163	156	123.613	1.262	36.3011	36.9007
TGTGGAGTTTG	92	61.8122	93	62.437	1.4895	37.055	36.5873

Table A.118: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AGGAAGAAGAAG	472	418.545	489	427.527	1.14379	65.6943	56.7316
CAGAAGAAGAAT	52	25.1042	52	25.3523	2.0511	37.3555	37.8667
CAGAAGAAGAAA	97	66.508	97	67.2454	1.44248	35.5371	36.6067
AGGAGGAAGAAG	171	139.092	176	140.929	1.24886	39.112	35.3167
GAGAAGAAGAAA	189	157.272	198	159.433	1.2419	42.8948	34.7324
AGGAAGAAGATG	153	122.421	154	123.978	1.24215	33.3946	34.1151
ATGAAGAAGAAG	352	319.874	367	325.802	1.12645	43.6993	33.6877
CAAGAAGAAGAT	111	82.3832	111	83.3348	1.33198	31.8197	33.0945
CGATGCTAAAGG	33	12.2809	33	12.3977	2.66179	32.307	32.6193
AAGAAGAAGATG	354	323.816	365	329.855	1.10655	36.9546	31.5494
GAGAAGAAGATT	84	58.1198	84	58.75	1.42979	30.0322	30.938
ACAACAACAATA	81	57.2352	81	57.8543	1.40007	27.2582	28.1297
TGAAGAAGAAGC	121	96.6558	121	97.8125	1.23706	25.7413	27.1808
TATTCCTGGATG	30	12.5538	30	12.6733	2.36719	25.8511	26.1352
TAAAGGAACAAT	40	21.0225	40	21.2277	1.88433	25.3429	25.7314
TTCTTCTTCCTC	252	228.069	256	231.677	1.10498	25.5569	25.1451
GAGGAGAAGAAA	129	106.323	134	107.626	1.24505	29.3699	24.9392
CATTGCAGAGAT	42	23.494	42	23.725	1.77028	23.9879	24.3989
GTTTCTCATCCG	40	21.7564	40	21.9692	1.82073	23.9695	24.3589
ACGAAGAAGAAG	219	195.994	226	198.91	1.13619	28.8558	24.3061
TAAATGCTCAAT	34	16.8178	34	16.9799	2.00237	23.6073	23.9335
ATTGCAGAGATT	47	28.3176	47	28.6	1.64336	23.3468	23.8133
CGAAAATTCCAT	34	16.9367	34	17.1	1.9883	23.3676	23.6939
TCTTCATCTTCT	153	131.168	155	132.87	1.16655	23.8779	23.5556
AGAGAAGAAGAG	131	109.495	135	110.847	1.2179	26.6121	23.4905

Table A.119: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAGAAGAAGAAGG	81	42.4282	81	42.9046	1.88791	51.4731	52.3775
GGAAGAAGAAGAG	103	74.8966	104	75.8085	1.37188	32.8828	32.8179
AAAGAAGAAGAAA	93	65.4673	95	66.2464	1.43404	34.2471	32.6475
GATGATGAAGAAG	152	124.565	155	126.263	1.2276	31.7843	30.2561
TGAAGAAGAAGAT	104	78.6418	105	79.6079	1.31896	29.0689	29.0667
GAGAAGAAGAAAG	91	71.2844	97	72.1448	1.34452	28.7154	22.2205
AGGAGAAGAAGAA	179	158.379	180	160.695	1.12013	20.4205	21.9081
GAAGAAGATGATG	148	127.934	152	129.69	1.17203	24.1275	21.5636
GAAGAAGAAGATG	254	233.416	264	237.345	1.11231	28.0992	21.4659
AGAAGAAGAAAAA	113	93.9194	115	95.1152	1.20906	21.832	20.8995
CAAGAAGAAGAAC	31	15.9941	31	16.1613	1.91816	20.1924	20.5148
GGAAGAGGAGGAG	67	49.38	68	49.9444	1.36151	20.9846	20.4448
GAAGCCAAAGACG	13	2.76554	13	2.79339	4.65385	19.99	20.1203
GAAGAAGAAGACG	119	100.765	119	102.068	1.16589	18.2651	19.7941
AAGAAGAAGAGGA	167	148.419	169	150.546	1.12258	19.5413	19.6979
AAGAAGCAGAAGA	76	58.735	76	59.4224	1.27898	18.7007	19.585
AGGAAGAAGAAGA	333	314.295	345	320.336	1.07699	25.5901	19.251
AGATGATGATGAC	42	26.7028	43	26.9904	1.59316	20.026	19.0218
GCAACAACAACAG	37	22.2838	37	22.5209	1.64292	18.3696	18.7611
AGAAGAAGAAGGA	145	127.454	151	129.202	1.16871	23.541	18.7014
GAGAAGAAGAAGC	62	45.9592	65	46.4799	1.39845	21.7988	18.5616
ATGAAGAAGAAGA	250	232.18	261	236.079	1.10556	26.1919	18.4867
AGATGATGAAGAA	77	60.8055	77	61.5208	1.25161	17.2812	18.1817
CGAGGAGAAGAAA	32	18.2547	33	18.4468	1.78893	19.1934	17.9621
CTGGTGAAAGAAT	17	5.92932	17	5.98958	2.83826	17.7343	17.9062

Table A.120: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTCTTCTTCTTCTC	42	28.1085	42	28.436	1.477	16.3806	16.8671
TTGAAGAAGAAGAA	83	68.2217	83	69.0966	1.20122	15.2168	16.2745
CAGAAGAAGAAGAG	23	11.5012	23	11.6296	1.97771	15.6846	15.9401
AGGAGGAAGAAGAA	79	65.3975	79	66.2308	1.1928	13.9279	14.9282
GGAAGAAGAAGATG	53	40.0139	53	40.494	1.30884	14.2643	14.8965
AAGAAGAGTGATGA	17	7.1623	17	7.24138	2.34762	14.5078	14.6945
AAGAAGAAGAAAAA	91	78.0925	93	79.1165	1.17548	15.036	13.9199
TTGATGATGATGAA	24	14.0065	24	14.164	1.69444	12.6565	12.9247
CAGAAGAAGAGAAA	23	13.1174	23	13.2645	1.73396	12.6593	12.9158
GAGGAAGAAGATGA	67	55.414	67	56.1039	1.19421	11.8917	12.7207
TATGAAGAAGAAGC	12	4.18845	12	4.23433	2.83398	12.5002	12.6309
GAGAAGAGTGATGT	23	13.3672	23	13.5172	1.70153	12.2251	12.4818
GGATGCAGAGGAAT	15	6.64671	15	6.72	2.23214	12.0444	12.2089
GTGCAGAGATTTCG	6	0.809395	6	0.818182	7.33333	11.9546	12.0194
GTGCAACTCCTCCC	28	18.4788	28	18.6889	1.49822	11.3197	11.6363
GACGTCTACAGCTT	5	0.506699	5	0.512195	9.7619	11.3924	11.4464
TAGAGAAGAGAAAC	9	2.52979	9	2.55738	3.51923	11.3242	11.4218
CAAGAAGAAGAAGC	24	14.9908	24	15.1597	1.58315	11.0259	11.2949
GAGCAAGAAGAAGA	29	19.7081	29	19.9329	1.45488	10.8728	11.2018
GTGATGAAGAAAGT	14	6.34567	14	6.41558	2.18219	10.9246	11.078
GAGAGGAAGAAGAG	18	9.77041	18	9.87903	1.82204	10.7992	10.9982
GTGAAGGAGAAGAT	17	8.94426	17	9.04348	1.87981	10.7299	10.9174
CCTGTTCTTGAAGA	24	15.2517	24	15.4237	1.55604	10.6115	10.8807
AGAAGAAGAAAGAG	60	50.0714	60	50.687	1.18374	10.1206	10.8537
GTGGCTACACCATT	5	0.570731	5	0.576923	8.66667	10.7974	10.8514

Table A.121: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ACCACCACCACCATC	17	6.15242	18	6.22535	2.8914	19.1114	17.2783
AGAAGAAGAAGCAGA	36	23.7527	36	24.0465	1.4971	14.527	14.9696
CGGAGATGAAGAGGC	15	6.07691	15	6.14894	2.43945	13.3766	13.5533
TTCCACCACCACCAC	12	3.89582	12	3.94175	3.04433	13.3594	13.5
AAAGCAAGAAGAAGA	13	4.82669	13	4.88372	2.6619	12.7275	12.8803
GGAAGAAGAAGAAGC	25	15.1126	25	15.2957	1.63444	12.2825	12.5836
AGAAGGAGAAGAAGA	59	47.9578	59	48.5849	1.21437	11.4592	12.2258
GAGAAGAAGAAGAAC	16	7.53014	16	7.61971	2.09982	11.8696	12.0588
GAGAAGAAGAAGAAA	36	25.7864	36	26.1069	1.37895	11.5675	12.0122
TCTCTCCTTCTCCTC	9	2.41607	9	2.44444	3.68182	11.7307	11.8358
CGAGGAGAAGAAAGT	21	12.3882	22	12.5373	1.75476	12.3713	11.0833
GTGAAGAAGAAGAAC	10	3.42568	10	3.46602	2.88515	10.5958	10.7128
AAGAAGAAGATGATG	47	37.4655	49	37.9439	1.29138	12.5298	10.6562
TGATGATGAAGAAGA	54	45.0406	54	45.6258	1.18354	9.09957	9.79661
ATGAAGAAGAAGAAG	111	101.745	115	103.236	1.11395	12.4099	9.66343
TGAAGAAGAAGAAGT	20	12.3884	20	12.5375	1.59522	9.34017	9.57948
GAGATGAAGAGGAAA	15	7.94116	15	8.03571	1.86667	9.36231	9.53986
CAGAAGAAGATGATA	7	1.81054	7	1.83178	3.82143	9.38437	9.46599
GATGATGAAGAAGAA	62	53.2797	62	53.9847	1.14847	8.58285	9.39792
CCAACAACAACAACT	9	3.21745	9	3.25532	2.76471	9.15241	9.25771
TTCTTCTTCTTCTCC	50	41.6509	50	42.1878	1.18518	8.49454	9.13505
ATGATGAAGAAGAAG	54	45.7455	54	46.3407	1.16528	8.25998	8.95811
ATACTTGAAGAAGAG	16	9.18068	16	9.29032	1.72222	8.69785	8.88781
AGAAGAAGATGATGA	51	42.9447	53	43.5	1.21839	10.4691	8.76752
TCAGCAGCAACAACT	7	2.00769	7	2.03125	3.44615	8.66081	8.74247

Table A.122: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CCTCCACCACCACCAC	16	5.8126	17	5.88636	2.88803	18.0298	16.201
AGATGATGATGATGAA	21	10.5585	21	10.694	1.96372	14.1717	14.4393
AAGGAAGAAGAAGAAA	17	7.80673	17	7.90625	2.1502	13.0145	13.2299
AAAGAAGAAGAAGAAA	26	15.918	26	16.1247	1.61243	12.4214	12.7568
CAAGAAGAAGAAGAAA	21	11.6655	21	11.8155	1.77733	12.0773	12.3457
GCTCCACCACCACCAC	9	2.28376	9	2.3125	3.89189	12.2301	12.3426
CCTCCTCCACCACCAC	16	7.84592	17	7.94595	2.13946	12.9294	11.4015
TTCCTTCTCCAAAGGG	6	0.960916	6	0.972973	6.16667	10.915	10.9898
GGAGGAAGAAGAAGAA	46	36.4432	46	36.9383	1.24532	10.0921	10.7128
ATCTTCTTCTTCTTCG	15	7.48982	15	7.58523	1.97753	10.2277	10.4176
GAGGAAGAAGAAGAAG	113	103.229	115	104.833	1.09698	10.6444	10.22
CGAAGAAGAAGAAGAT	21	13.161	21	13.3309	1.57529	9.54326	9.81252
GCAACAACAACAACAG	16	8.80782	16	8.92035	1.79365	9.34805	9.55119
TGAAGAAGAAGAAGAG	28	20.0598	28	20.3227	1.37777	8.97308	9.33767
CGATGATGATGATGAA	13	6.48986	13	6.57234	1.97799	8.86703	9.03121
TTGAAGAAGAAGAAGT	11	4.84771	11	4.90909	2.24074	8.87487	9.01327
GATGATGATGATGAAG	59	50.993	59	51.7075	1.14103	7.78409	8.60509
TGGAGGAGGAGGAGGT	12	6.01223	12	6.08856	1.97091	8.14194	8.29332
CAGAAGAAGAAGAAGT	6	1.53895	6	1.55829	3.85038	8.08903	8.16395
TTCATCATCATCATCT	10	4.47173	10	4.5283	2.20833	7.92238	8.04809
TGAAGAAGAAGAAGAC	18	11.5562	18	11.7048	1.53783	7.74666	7.97662
ATGGTGGTGGTGGTGA	7	2.30433	7	2.33333	3	7.69029	7.77784
TGAAGATGAAGAAGAT	12	6.29814	12	6.37815	1.88142	7.58434	7.73582
CCCCCACCACCACCAC	4	0.589623	4	0.597015	6.7	7.60843	7.65827
AGAAGAAGATGAAGAA	37	30.0967	37	30.5	1.21311	7.14808	7.64059

Table A.123: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CGAGGAAGAAGAAGAAA	7	1.50917	7	1.52941	4.57692	10.6472	10.7404
TGAGGAGGAGGAGGTGC	5	0.822323	5	0.833333	6	8.9588	9.0253
GGAAGAAGAAGAAGAAA	18	11.992	18	12.1565	1.48069	7.06516	7.31038
ACCACCACCACCTCCAC	9	4.11689	9	4.17241	2.15702	6.91857	7.03913
CTTATGTTCCAAGTTTA	18	12.2133	18	12.381	1.45385	6.73583	6.98116
CAAGAAGAAGAAGAAGT	7	2.59219	7	2.62703	2.66461	6.8604	6.95386
AGTGGTGGTGGTGGTGT	3	0.296041	3	0.3	10	6.90776	6.94761
GGCTGCTGCTGCTGCTA	4	0.730956	4	0.740741	5.4	6.7456	6.79879
AAGGAGGGAGAGTTTTG	5	1.29839	5	1.31579	3.8	6.67501	6.74158
TCTCTTCTTCTTCTTCC	13	7.90236	13	8.0098	1.62301	6.29568	6.47125
GGAAGAAGAAGAGCAAA	5	1.43904	5	1.45833	3.42857	6.16072	6.22731
CGATGATGAAGAAGAAC	3	0.381987	3	0.387097	7.75	6.14308	6.18294
TATGATGATGATGATGG	4	0.87877	4	0.890538	4.49167	6.0089	6.0621
AGGAGAAGAAGAAGAAG	40	34.429	41	34.9239	1.17398	6.57641	5.99919
GTGATGATGATGATGAG	8	3.80881	8	3.86014	2.07246	5.82991	5.937
AGGAGGAGGAGGAGGAA	25	19.7444	25	20.0197	1.24877	5.55397	5.90015
GAAGAAGATGAAGAAGA	39	33.6549	39	34.1379	1.14242	5.19295	5.74868
TTGAAGAAGAAGAAGAC	8	3.90117	8	3.95376	2.02339	5.6382	5.74532
TAAGAAGAAGAAGAAGC	6	2.30418	6	2.33514	2.56944	5.66214	5.7422
CGGAGAAGAAGAAGAAA	7	3.08887	7	3.13043	2.23611	5.63317	5.72673
CTCTTCTTCTTCTTCCT	17	12.2005	17	12.3679	1.37452	5.40782	5.63952
AGATGATGATGATGATA	9	4.81912	9	4.88421	1.84267	5.50095	5.6217
GTGAAGAAGAAGAAGAG	8	3.99242	8	4.04624	1.97714	5.45322	5.56036
TAACAACAACAACAACG	6	2.39095	6	2.42308	2.47619	5.44033	5.5204
TTGGTGGTGGAGGAGGT	3	0.477483	3	0.483871	6.2	5.47365	5.51352

Table A.124: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTGATGATGATGATGATC	7	1.03037	7	1.04505	6.69828	13.313	13.4119
AAGGAGGAAGAAGAAGAC	6	1.39193	6	1.41176	4.25	8.68151	8.76643
AAAGAAGAAGAAGAAGAA	52	45.3802	52	46.0856	1.12834	6.27864	7.08075
GCTTCTTCTTCTTCCTCA	5	1.29659	5	1.31507	3.80208	6.67775	6.7485
CGAAGAAGAAGAAGAGAG	4	0.80214	4	0.813559	4.91667	6.37052	6.42707
CAAGAAGAAGAAGAAGTG	6	2.14713	6	2.17778	2.7551	6.08073	6.16577
AGAAGAAGAAGAAGAGAA	23	17.6051	23	17.8644	1.28748	5.81173	6.14803
CAATACTTGAAGAAGAGG	7	3.01932	7	3.0625	2.28571	5.78675	5.88615
GAAAAGAAGAAGAAGAAA	5	1.62982	5	1.65306	3.02469	5.53405	5.60484
GTGGATTACCTGATATAG	2	0.127224	2	0.129032	15.5	5.48168	5.50991
CCGTCTTCTTCTTCTTCA	3	0.508889	3	0.516129	5.8125	5.28003	5.32241
GAATACTTGAAGAAGAGT	9	4.99084	9	5.0625	1.77778	5.17828	5.30659
ACTTCTTCTTCTTCTCCC	2	0.140855	2	0.142857	14	5.27811	5.30635
GCAACAACAACAACAACC	7	3.30159	7	3.34884	2.09028	5.16108	5.26053
CTGATGAAGAAGAAGATA	3	0.521985	3	0.529412	5.66667	5.2038	5.24619
TTCTTCTTCTTCTTCCTC	33	28.1563	33	28.5797	1.15467	4.74575	5.23833
GGAAGAAGAAGAAGAGAT	5	1.75463	5	1.77966	2.80952	5.16508	5.23589
AAAGAAGAAGAAGAAGCC	4	1.08455	4	1.1	3.63636	5.16394	5.22051
TCTTCTCCTCCTCCTCCT	9	5.11177	9	5.18519	1.73571	4.96277	5.09111
ACTTCTTCTTCTTCTTCA	6	2.57534	6	2.61214	2.29697	4.98954	5.07467
GGATGATGAAGAAGATGG	3	0.591582	3	0.6	5	4.82831	4.8707
TAAGAAGAAGAAGAAGTC	4	1.20505	4	1.22222	3.27273	4.74249	4.79908
TATCTTCTTCTTCTTCTG	2	0.183438	2	0.186047	10.75	4.74981	4.77805
GTGATGATGATGATGATA	6	2.71791	6	2.75676	2.17647	4.66623	4.75137
GAGAAGAAGAAGAAGGAA	5	2.02242	5	2.05128	2.4375	4.45486	4.52572

Table A.125: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GGAAGAAGAAGAAGAAGAT	13	6.66656	13	6.76827	1.92073	8.48514	8.68198
TCAACAACAACAACAACAT	6	1.73016	6	1.7563	3.41627	7.37129	7.46129
AATGATGATGATGATGATA	6	2.02926	6	2.05995	2.9127	6.41448	6.50452
GAACAACAACAACAACAAA	4	0.791009	4	0.802941	4.98168	6.42307	6.48296
TGAAGAAGAAGAGGAAGAG	6	2.12176	6	2.15385	2.78571	6.14703	6.23709
ACTTGTTGCAAAGCATTTC	3	0.39406	3	0.4	7.5	6.04471	6.08959
CGGATGATGATGATGATGT	2	0.127117	2	0.129032	15.5	5.48168	5.51158
ACTTCTTCTTCTTCTTCTA	3	0.496713	3	0.504202	5.95	5.35017	5.39506
TGTTCCAACTGATTCCTGT	2	0.135884	2	0.137931	14.5	5.3483	5.3782
TAAGAAGAAGAAGAAGAAC	3	0.504801	3	0.512411	5.85467	5.30172	5.34661
AGGTGGTGGTGGTGGTGGC	4	1.11552	4	1.13235	3.53247	5.04799	5.10791
CTTCTTCATTGACGAGGAT	5	1.81468	5	1.84211	2.71429	4.99264	5.06765
GAGAAGAGGAAGAAGAAGG	3	0.562941	3	0.571429	5.25	4.97468	5.01958
TGTTAAAATCACTCCTAAG	7	3.44772	7	3.5	2	4.85203	4.95737
AGAAGAAGAAGAAGAAGCA	13	9.09965	13	9.23913	1.40706	4.43952	4.63728
CCATCATCATCATCATCAT	12	8.15591	12	8.2807	1.44915	4.45175	4.63396
CACAACAACAACAACAACC	3	0.664972	3	0.675	4.44444	4.47496	4.51987
TGGAGGAAGAAGAAGAAGC	3	0.672781	3	0.682927	4.39286	4.43994	4.48484
ACACTGTGGATGGAACTCG	4	1.3135	4	1.33333	3	4.39445	4.4544
GGAAAGAAGAAGAAGAAGG	3	0.695395	3	0.705882	4.25	4.34076	4.38566
AGGAGGAGGAGGAGGTGGA	4	1.39076	4	1.41176	2.83333	4.16582	4.22577
CGGGACAAAGAAGGTGTAC	2	0.246289	2	0.25	8	4.15888	4.18879
TCACCACCACCACCACCAT	4	1.42295	4	1.44444	2.76923	4.07428	4.13424
GGAAGAAGAAGAAGAAGTG	4	1.42673	4	1.44828	2.7619	4.06368	4.12364
CGATGAAGAAGAAGAAGAG	3	0.770979	3	0.782609	3.83333	4.0312	4.07612

Table A.126: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AGATGATGATGATGATGATT	5	1.18117	5	1.2	4.16667	7.13558	7.21468
CGAAGAAGAAGAAGAAGAAA	7	2.56537	8	2.60638	3.06939	8.97182	7.02665
GAGAAGAAGAAGAAGAAGAT	7	3.04883	7	3.09761	2.2598	5.70695	5.81807
CTCTGTTTTGATTTTACAAC	7	3.44484	7	3.5	2	4.85203	4.96323
AAAGAAGAAGAAGAAGAAGC	5	2.02205	5	2.05435	2.43386	4.4474	4.52662
CAGAAGGAGAAGAAGAAGAG	2	0.231607	2	0.235294	8.5	4.28013	4.31172
ATGGTTTGGTCCTGGAAGTC	5	2.15311	5	2.1875	2.28571	4.13339	4.21263
ATCTGTTTTGATTTTACAAG	7	3.9369	7	4	1.75	3.91731	4.02861
TTTCTTCTTCTTCTTCCTCG	2	0.268454	2	0.272727	7.33333	3.98486	4.01645
CTCATCATCATCATCATCAC	3	0.787453	3	0.8	3.75	3.96527	4.01269
AGAAGAAGAAGAAGAAGATG	23	19.3376	23	19.6562	1.17011	3.61327	3.98923
CGAAGAAGAAGAAGAAGAGG	5	2.26383	5	2.3	2.17391	3.88264	3.9619
GTCATCATCATCATCATCAA	3	0.801514	3	0.814286	3.68421	3.91217	3.9596
GGAGGAAGAAGAAGAAGAAG	22	18.4374	22	18.7407	1.17391	3.52754	3.88657
CGGAAGAAGAAGAAGAAGAG	2	0.289509	2	0.294118	6.8	3.83385	3.86543
TAGCAGCAGCAGCAGCAGCT	2	0.295299	2	0.3	6.66667	3.79424	3.82583
CGTTGATGATGATGATGATC	3	0.843698	3	0.857143	3.5	3.75829	3.80572
ACTCTTCTTCTTCTTCTTCC	3	0.843698	3	0.857143	3.5	3.75829	3.80572
CAGAAATTCCAGATCTTTCA	4	1.55415	4	1.57895	2.53333	3.71814	3.78147
GTCCCCACCACCACCATACG	3	0.851298	3	0.864865	3.46875	3.73138	3.77881
CCTACACCATGAAAGCTTTA	2	0.302871	4	0.307692	13	10.2598	3.77519
AAGCAACAACAACAACAACC	2	0.310841	2	0.315789	6.33333	3.69165	3.72324
GAGAAGAAGAAGAAGAAGCG	2	0.314985	2	0.32	6.25	3.66516	3.69675
GGACTCATATGGACTTTGGG	1	0.0252394	1	0.025641	39	3.66356	3.67935
TTTCTTCTTCTTCTTCTTCC	5	2.42282	5	2.46154	2.03125	3.54326	3.62254

A.7 The *Arabidopsis thaliana* Intron Segments

A.7.1 Timing Information

Table A.127: Full Size and Timing Information, Intron Segments

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	2.366837	3.424127	33.391794	5.790964
2	0	0.000001	4.256545	3.879843	60.157790	8.136388
3	1	0.000003	6.489944	3.846629	37.109455	10.336573
4	2	0.000012	9.903832	3.489371	16.538488	13.393203
5	3	0.000046	13.368255	5.111136	10.636120	18.479391
6	4	0.000186	16.838000	8.426837	6.760218	25.264837
7	5	0.000743	20.737331	31.773659	6.085598	52.51099
8	6	0.002969	26.261253	98.301754	6.008992	124.563007
9	7	0.011734	33.631734	278.397045	7.631363	312.028779
10	8	0.043770	44.300981	1288.922963	12.824636	1333.223944
11	9	0.143836	53.926168	2967.204012	27.642505	3021.13018
12	10	0.394024	64.624168	6529.431648	58.883640	6594.055816
13	11	0.879558	77.275459	12883.104469	103.556977	12960.379928
14	12	1.617681	88.921866	19423.546231	151.086124	19512.468097
15	13	2.540684	100.765473	15564.251207	188.379718	15665.01668
16	14	3.560202	113.781616	17037.425231	217.439355	17151.206847
17	15	4.617928	127.728452	24603.436737	241.977325	24731.165189
18	16	5.686391	139.826736	21624.955253	265.337579	21764.781989
19	17	6.754656	150.393343	23493.679380	284.917113	23644.072723
20	18	7.818624	164.598093	17941.167227	309.049341	18105.76532

A.7.2 Top 25 Words, $S * ln(\frac{S}{E_s})$

Table A.128: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 1

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
C	116467	116475	2948210	2.94821e+06	1	0	-8.01561
G	116477	116476	3223156	3.22316e+06	1	0	1.34939
A	116478	116477	5245458	5.24546e+06	1	1.16473e-09	0.903381
T	116477	116478	7667433	7.66743e+06	1	0	-0.539004

Table A.129: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 2

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CG	84349	108576	252212	497926	0.506525	-171550	-21296.9
AC	114113	114616	778036	810339	0.960137	-31650.1	-501.651
CC	99690	106809	425684	455451	0.934643	-28772.2	-6876.38
AG	116475	115154	909600	885910	1.02674	24004.3	1328.65
GC	105407	108576	448377	497926	0.90049	-46997.2	-3122.13
GG	97577	110129	444770	544361	0.817049	-89868.5	-11808.1
AT	116429	116465	1882724	2.10745e+06	0.893364	-212298	-35.8013
CA	115633	114616	960778	810339	1.18565	163612	1021.75
GT	116475	116255	1281314	1.29496e+06	0.989463	-13572.9	220.263
TC	116119	116122	1296108	1.18449e+06	1.09423	116714	-2.75168
TA	116417	116465	1677218	2.10745e+06	0.79585	-382983	-47.797
GA	114516	115154	932300	885910	1.05236	47584.2	-636.127
CT	116171	116122	1309468	1.18449e+06	1.10551	131346	49.2587
AA	115925	116356	1675090	1.44176e+06	1.16184	251273	-430.029
TG	116319	116255	1500181	1.29496e+06	1.15848	220687	64.0723
TT	116444	116475	3193916	3.08052e+06	1.03681	115453	-30.8465

Table A.130: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CAG	94475	75229	186351	166606	1.11851	20871.6	21521.2
GTA	106160	93425.7	318153	280282	1.13512	40321.6	13565.3
GCA	75889	70421.3	159307	146119	1.09025	13765.5	5674.71
TGC	87488	83386.7	224628	208692	1.07636	16529.6	4200.59
AAC	92939	89452.4	273789	248459	1.10195	26579.6	3553.65
CTC	88196	85468.9	255129	221353	1.15259	36230.6	2770.15
TGG	85161	83098.9	238952	207013	1.15428	34285	2087.47
TAC	91153	89495.3	251238	248774	1.0099	2475.7	1672.91
CTT	111614	109976	563555	545467	1.03316	18384.8	1649.82
GTT	111022	109612	560643	533739	1.05041	27570.8	1418.76
TTC	111132	109806	555150	539902	1.02824	15461.6	1333.85
ACC	62197	60918.4	121291	112338	1.07969	9300.11	1291.93
ATC	98523	97356.9	320988	318257	1.00858	2742.76	1173.04
TGA	106574	105512	450114	433928	1.0373	16483.8	1067.3
TTG	112992	111971	653770	624910	1.04618	29516.9	1025.29
ACA	91127	90135.6	274418	253550	1.0823	21703.7	996.805
CCA	69060	68522.9	147343	138724	1.06213	8881.25	539.162
TCT	111155	110825	619499	575675	1.07613	45450.9	330.495
TAA	109987	109672	499363	535605	0.932335	-34986.8	315.9
ATG	101734	101482	381064	368367	1.03447	12913.7	252.142
CAT	99821	99678.2	347743	344847	1.0084	2908.25	142.934
TCC	78842	79467.7	183747	187141	0.981861	-3363.52	-623.262
CCG	27220	27851.4	39376	36416.2	1.08128	3076.95	-624.163
ATT	113605	114292	709181	784260	0.904268	-71364.9	-684.709
TCA	104167	104895	379709	422382	0.89897	-40441.3	-725.191

Table A.131: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GCAG	40283	24257.4	47592	30898.9	1.54025	20557	20431.9
GTAA	70137	54675.7	112595	94724.6	1.18866	19459	17466.3
TGCA	56593	49008.2	88770	79809.7	1.11227	9445.51	8143.61
GTTT	94106	87745.4	268075	238820	1.1225	30977.4	6585.74
TTAC	60221	54249.7	104032	93541	1.11215	11058.5	6288.52
TTTC	93418	87406.5	253160	236480	1.07053	17254.4	6213.62
TAAG	55023	49753.7	88916	81673.5	1.08868	7554.52	5538.94
CTAA	53329	48411.6	86983	78338.3	1.11035	9105.05	5159.07
CTTA	64566	59906.4	115617	110184	1.04931	5564.72	4836.3
TTTG	97169	92875.3	300450	278490	1.07885	22803.7	4391.42
AATC	58254	54113.5	103858	93165	1.11478	11284.5	4294.98
AAGC	32801	28875.5	45604	38419.4	1.18701	7818.08	4181.01
GGTT	52439	48720.1	93218	79096.9	1.17853	15312.6	3857.35
CTGA	47707	44144.6	72812	68309.5	1.06591	4647.77	3702.41
TGTT	93600	90184.6	269282	256619	1.04935	12970.6	3479.26
AGTC	32107	28862.5	44174	38397.3	1.15045	6190.95	3420.4
AAGT	56223	52913.7	99518	89897.8	1.10701	10117.5	3410.67
TTGC	59081	55796.9	99367	97891.6	1.01507	1486.51	3378.89
TATG	68417	65152.7	134183	127640	1.05127	6708.4	3344.78
GTGA	48623	45460.5	76608	71312.6	1.07426	5487.36	3270.06
ACTA	44979	41864.4	73764	63283.8	1.16561	11303.7	3227.73
CATT	69161	66088.3	134288	130987	1.0252	3342.05	3143.01
TTCC	51577	48560	83227	78702.7	1.05749	4651.96	3108.82
CCTT	49829	47115.2	77957	75201	1.03665	2805.86	2790.51
TAAC	52363	49732.3	82639	81619.6	1.01249	1025.79	2699.08

Table A.132: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTAAG	26230	16750.2	28598	20048.5	1.42644	10157.5	11764
TGCAG	29602	21251.5	32079	26519.5	1.20964	6105.32	9810.57
TGTAG	22772	17305.6	27575	20819.7	1.32447	7748.89	6251.01
TTCAG	27531	22817.2	32703	28892.8	1.13187	4051.03	5170.24
TTTTG	67751	63504.8	129061	123091	1.0485	6112.79	4385.1
GTATG	22748	18811.4	26706	22948.6	1.16373	4049.49	4322.4
TTCTT	64181	60229.1	120077	112245	1.06978	8099.26	4078.79
CTTTT	58270	54374.1	95839	94743.2	1.01157	1102.13	4032.29
GTGAG	14256	10749.3	15772	12176.2	1.29531	4081.02	4024.95
GTTTC	38683	35038.7	53362	49881.9	1.06977	3598.77	3827.58
TTTCT	66239	62658.3	120557	120208	1.0029	349.029	3681.07
TATGT	39569	36398.1	55934	52518	1.06504	3524.75	3305.15
TCAGT	16604	13651.3	19581	15880.3	1.23303	4101.79	3251.24
TGTAT	32855	29857.6	44708	40417.3	1.10616	4510.81	3143.03
TAAGT	27928	24990.9	34234	32298.2	1.05993	1992.67	3103.3
TTTTC	60414	57470	102054	103717	0.98397	-1649.18	3018.11
TGTTT	68035	65121.4	127168	128759	0.987644	-1581.02	2977.82
CCTTT	27617	24797.5	34765	31990	1.08675	2892.07	2973.99
ACTTG	23581	20852.9	29020	25925.7	1.11935	3272.06	2899.28
TTTGT	64622	61789.5	118408	117309	1.00937	1104.5	2896.49
TCTTT	61778	58956.8	106372	108247	0.98268	-1858.52	2887.61
GTTTT	62091	59464.1	110267	109827	1.00401	440.814	2684.12
TCTAT	30845	28313.8	40727	37764.7	1.07844	3075.53	2641.15
ATTCA	28019	25507.2	36667	33126.5	1.10688	3723.35	2631.64
CATTT	41320	38890.2	59491	57527.4	1.03413	1996.7	2504.18

Table A.133: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTAAGT	13240	9743.29	13661	11010.7	1.24071	2946.42	4060.23
TTGCAG	15359	12102.5	15921	13977.5	1.13904	2072.74	3659.93
TTTCAG	15849	12627.6	17220	14654.8	1.17504	2777.63	3601.24
GTTTTG	23453	20458.4	29423	25531.7	1.15241	4173.87	3203.8
TGTTTT	38710	36070.6	53508	52307.9	1.02294	1213.79	2733.67
TTTTGT	37896	35332.5	52939	50863.2	1.04081	2117.57	2654.32
TTTTCT	36498	34154.9	49294	48599	1.0143	699.941	2421.69
TGTAAT	10606	8538.69	11798	9542.7	1.23634	2502.98	2299.51
ATTTTG	24651	22482.8	31155	28595.9	1.08949	2670.32	2269.49
CTCTTT	19154	17032.4	22198	20588.6	1.07817	1670.77	2248.59
TTTCTT	39400	37378.2	55343	54916.4	1.00777	428.271	2075.56
ATTTTC	20405	18459.8	24333	22612	1.07611	1784.89	2044.22
CCTTTT	14317	12436.4	16123	14407.5	1.11907	1813.77	2016.08
TTTGTT	39883	38060.4	57820	56302.9	1.02694	1537.31	1865.6
GTGAGT	7244	5698.85	7572	6203.69	1.22056	1509.2	1737.9
CTGTTT	17700	16111.8	20906	19310.2	1.08264	1659.95	1664.04
CATTTT	21144	19544.6	25379	24183.9	1.04942	1224.13	1663.15
ATTTCA	14663	13097.6	16861	15266.4	1.10445	1675.12	1655.37
TGTGAT	12203	10657.7	13805	12146	1.13658	1767.43	1652.25
CTCTGT	10822	9314.01	12385	10483.9	1.18133	2063.85	1623.97
AGCAGA	1935	882.423	2012	917.996	2.19173	1578.8	1519.35
TGTTTC	21746	20311.5	25503	25313.6	1.00748	190.124	1483.98
TCTTTC	18028	16609.1	21366	19998.3	1.06839	1413.49	1477.84
GTATGT	11224	9842.01	12147	11132.4	1.09114	1059.54	1474.77
TGTTAT	16127	14728.1	18467	17427.2	1.05966	1070.18	1463.37

Table A.134: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTG	9908	7180.24	10899	7978.19	1.3661	3400.05	3190.48
TTTTTGT	17751	15641.4	20722	18800.9	1.10218	2016.1	2245.82
TTTTCTT	20203	18355.3	24108	22628.9	1.06536	1526.37	1937.72
CTTTTTC	7403	5851.68	7918	6421.9	1.23297	1658.22	1740.86
TTTGTTT	22807	21217.8	28905	26870.7	1.07571	2109.48	1647.27
ATTTTTA	7741	6304.28	8751	6947.91	1.25951	2019.09	1589.25
GATTTTG	7541	6216.64	8383	6845.72	1.22456	1698.25	1456.36
TTTTGTT	21930	20542.6	26940	25850.7	1.04214	1111.91	1433.18
TGTTTTT	17299	15959.4	20183	19240	1.04901	965.725	1394.26
ATTTTTG	9159	7918.65	9972	8859.33	1.12559	1179.78	1332.79
ATTTTGA	8150	7004.42	8977	7770.1	1.15533	1296.12	1234.53
TTTTTCT	16894	15740.7	19492	18937.7	1.02927	562.316	1194.52
TTTCTTT	18554	17424.1	22544	21294.8	1.05866	1285.17	1165.72
ATTTTTC	7920	6841.81	8511	7578.22	1.12309	987.962	1159
GTTTTTA	6751	5708.92	7166	6256.88	1.1453	972.185	1131.87
TTCTTTT	16739	15645.6	19438	18806.6	1.03357	641.865	1130.73
GGTTTTG	5839	4858.44	6267	5282.52	1.18637	1071	1073.45
CATTTTC	5562	4609.67	5835	5000.34	1.16692	900.737	1044.55
GTTCTTG	4224	3308.91	4457	3545.6	1.25705	1019.62	1031.35
GTTGTTG	4677	3756.09	5710	4041.82	1.41273	1972.95	1025.56
TTTGCAG	7319	6367.56	7426	7021.8	1.05756	415.613	1019.22
TGTTTTC	9723	8802.85	10410	9929.84	1.04835	491.581	966.647
TTCTCTT	10035	9122.25	10946	10320.7	1.06059	643.868	956.965
GTGTTTG	4619	3759.4	4878	4045.5	1.20578	912.815	951.134
TTATATT	7087	6239.54	7788	6872.41	1.13323	974.038	902.569

Table A.135: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTGTT	9926	9245.61	10963	10545.2	1.03962	426.002	704.837
TTTTTCTT	9086	8422.61	9952	9532.87	1.04397	428.215	688.857
CTTTTTTC	2742	2154.45	2799	2298.27	1.21787	551.703	661.242
GTTTTTGA	2643	2073.32	2711	2210	1.2267	553.926	641.623
TTTTGCAG	3479	2936.05	3497	3155.58	1.10819	359.256	590.319
TTTTTTGT	7536	6976.29	8115	7789.68	1.04176	332.019	581.582
TTTTTTGG	3720	3199.6	3897	3447.5	1.13038	477.603	560.592
TTTTCTTT	9179	8657.61	10214	9820.41	1.04008	401.378	536.781
CTCTCTTT	3188	2696.87	3284	2891.9	1.13558	417.553	533.357
ATTTTTTA	2453	1990.54	2583	2120.07	1.21836	510.153	512.452
TGTTTTTT	7356	6867.66	7888	7660.57	1.02969	230.772	505.308
TTTTTTCC	3125	2662.43	3209	2854.03	1.12437	376.18	500.608
GGTTTTTG	2012	1605.35	2075	1703.48	1.2181	409.378	454.284
TTTTGTTT	12001	11555.9	13731	13467.7	1.01955	265.887	453.541
TTTGTTTT	10866	10422.8	12366	12019.2	1.02885	351.76	452.544
TGTTTCAG	2160	1753.27	2184	1863.1	1.17224	347.078	450.632
CTTTTTTA	2190	1794.44	2236	1907.61	1.17215	355.166	436.264
AATATATT	1976	1595.71	2092	1693.09	1.23561	442.592	422.377
TTTTTTCT	7620	7215.87	8193	8075.31	1.01457	118.547	415.244
ATTTTTCA	2365	1986.26	2420	2115.43	1.14398	325.512	412.745
ATTTTTTC	2765	2389.01	2833	2554.24	1.10914	293.449	404.132
CAATTTTT	2357	1985.92	2434	2115.06	1.1508	341.867	403.772
CTGTTTTT	3659	3288.93	3828	3546.77	1.07929	292.092	390.146
TTTCATTT	4593	4219.52	4799	4590.71	1.04537	212.942	389.545
TTTTTAAT	3715	3348.02	3943	3612.52	1.09148	345.155	386.4

Table A.136: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTCTTT	4336	3862.83	4592	4216.43	1.08907	391.819	501.033
TTCTCTCTT	1423	1033.05	1451	1097.21	1.32245	405.531	455.712
TTTTTTTGT	4383	3969.07	4615	4336.79	1.06415	286.947	434.804
TTTTTTCTT	4232	3833.36	4435	4183.09	1.06022	259.35	418.682
TTATATATT	1143	804.184	1163	852.209	1.36469	361.607	401.86
ATTTTTTTA	1229	891.144	1261	945.169	1.33415	363.542	395.063
CTTTTTTTC	1301	961.718	1321	1020.73	1.29417	340.65	393.12
ATTTTCTTA	1242	949.62	1258	1007.77	1.2483	279.005	333.373
GATTTTTTT	2026	1720.8	2080	1839.97	1.13045	255.043	330.789
TTTTTTTCT	4112	3796.27	4307	4141.14	1.04005	169.138	328.512
TTTTCTTTT	4319	4012.99	4719	4386.62	1.07577	344.667	317.396
GTTTTTTTC	1341	1061.12	1361	1127.33	1.20728	256.371	313.914
TTTTTGTTT	5365	5061.01	5741	5587.71	1.02743	155.372	312.94
ATTTTTTTC	1364	1091.15	1387	1159.57	1.19613	248.405	304.433
TCTTTTTTT	3483	3197.62	3634	3468.16	1.04782	169.744	297.755
ATTTTTTTG	1476	1222.22	1504	1300.52	1.15646	218.622	278.474
TTTGTTTTT	4460	4191.88	4744	4590	1.03355	156.557	276.519
AGTTTTTTT	2099	1841.8	2157	1971.67	1.094	193.785	274.376
TAAAAAAAT	774	548.031	797	579.297	1.37581	254.275	267.216
TTATATATC	614	407.378	621	430.022	1.44411	228.215	251.896
TTTTTTGTT	4175	3930.66	4382	4293.25	1.02067	89.6602	251.782
TTTTCATTT	1947	1724.12	1976	1843.58	1.07183	137.071	236.706
ATTTTCTTG	1102	889.577	1116	943.493	1.18284	187.396	235.978
CTTCTTTTT	1752	1535.15	1772	1638.5	1.08148	138.799	231.494
TTTTGTTTT	5533	5308.15	6134	5874.33	1.0442	265.325	229.551

Table A.137: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTA	608	406.863	614	432.262	1.42043	215.491	244.233
GATTTTTTTT	1203	990.289	1222	1058.21	1.15478	175.857	234.076
TTTTTGTGTG	604	436.816	607	464.224	1.30756	162.774	195.733
GTTTTTTTTT	2930	2742.44	3048	2981.18	1.02241	67.5632	193.831
TTTTTTTTGT	2614	2431.12	2710	2634.78	1.02855	76.2836	189.593
AGTTTTTTTT	1313	1148.85	1338	1229.57	1.08818	113.075	175.353
TTTTTCTTTC	1072	922.838	1079	985.477	1.0949	97.8264	160.616
TTATATATAA	303	180.418	308	191.249	1.61047	146.769	157.093
TAAAAAAAAT	346	224.471	356	238.052	1.49547	143.27	149.711
CTTTTTTTTT	2498	2353.76	2576	2549.02	1.01058	27.1187	148.57
ATTTTTTTTG	728	594.348	735	632.629	1.16182	110.24	147.665
TTGTTTTTGT	1153	1014.41	1174	1084.25	1.08278	93.3682	147.65
TTTTTTCTTT	2038	1898.03	2101	2046.37	1.02669	55.3481	145.006
TTTTTTTTTG	3209	3068.23	3354	3345.87	1.00243	8.13799	143.95
TTTTTTTCTT	2295	2156.98	2365	2331.45	1.01439	33.7933	142.342
ATTTTTTTTC	595	468.759	596	498.329	1.196	106.672	141.891
TTTCTTCTTT	1058	930.316	1068	993.537	1.07495	77.1866	136.071
ATTTTTCTTA	436	319.183	438	338.813	1.29275	112.466	135.979
TTTTTTTTAA	1223	1094.41	1256	1170.67	1.07289	88.3624	135.867
TTTTTTTAAT	989	862.126	1014	920.093	1.10206	98.5444	135.782
CTTTTCTTTT	953	828.855	988	884.293	1.11728	109.564	133.01
TTTTTTTTCT	2141	2012.65	2210	2172.39	1.01731	37.938	132.353
TTTTCTTCTT	1284	1160.13	1304	1241.78	1.0501	63.7492	130.255
CTCTCTTTTT	633	517.434	637	550.341	1.15746	93.1493	127.605
GTTTTTGTTT	1338	1216.55	1380	1302.89	1.05918	79.3457	127.322

Table A.138: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTTTTTTTTA	333	207.643	333	221.603	1.50269	135.616	157.283
TTTTTTTCTTT	1162	1041.09	1185	1120.38	1.05768	66.4536	127.674
TTTTTTTTTAA	809	697.714	827	748.285	1.10519	82.7171	119.724
TTTTTTCTTTC	566	461.404	568	493.68	1.15054	79.6527	115.645
GTATATATATT	171	90.5322	171	96.5048	1.77193	97.8241	108.749
GATTTTTTTTT	723	624.484	731	669.258	1.09225	64.5061	105.908
TTTTTGTTTTT	1018	919.403	1078	988.225	1.09085	93.735	103.704
TTATATATATT	174	97.7776	174	104.236	1.66929	89.1575	100.287
TTTTTCTTCTT	608	516.815	613	553.274	1.10795	62.84	98.7937
TTTTTTTTCTT	1217	1126.69	1236	1213.52	1.01852	22.6824	93.837
ATTTTTTTTTG	408	325.054	412	347.316	1.18624	70.3643	92.7288
CTCTTTCTCTC	224	149.113	229	159.044	1.43985	83.4802	91.1554
TTTTGTTGTTG	464	381.694	467	408.068	1.14442	62.9963	90.6028
TTTTTCTTTCT	633	548.834	635	587.74	1.08041	49.1111	90.3124
TTATATATATG	141	75.5723	141	80.5459	1.75055	78.9505	87.9374
GTATATATATG	134	69.9712	134	74.5719	1.79692	78.5342	87.0674
TTTTTATTTTT	625	543.749	669	582.265	1.14896	92.8967	87.0397
TTGTGTGTGTT	219	147.65	220	157.481	1.397	73.5512	86.3365
CTGTTTTTTTT	540	461.287	545	493.554	1.10424	54.0386	85.076
TTTTTTGTGTG	299	225.565	300	240.773	1.24599	65.9782	84.2686
ATTTTTTTTTC	315	241.134	317	257.433	1.23139	65.9817	84.1736
TTTTTGTTTGT	583	506.604	586	542.287	1.08061	45.4294	81.8865
TCTTTTTTTTT	1220	1143.98	1241	1232.36	1.00701	8.66665	78.4873
AATATATATAA	108	52.3122	109	55.7419	1.95544	73.097	78.2893
TTTTTGTTGTT	538	465.797	540	498.401	1.08346	43.2882	77.5308

Table A.139: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTC	326	181.452	326	194.869	1.67292	167.75	191.006
CTTTTTTTTTTC	244	150.34	245	161.406	1.51792	102.248	118.162
ATTTTTTTTTTC	207	120.7	207	129.545	1.5979	97.0185	111.658
CTTCTTCTTTTT	270	194.63	271	209.049	1.29634	70.3377	88.3773
TTTTTGTTGTTG	270	197.625	271	212.273	1.27666	66.1909	84.254
AGTTTTTTTTTT	600	526.983	607	567.93	1.06879	40.3843	77.8572
TTTTTTTTTTGA	305	236.692	306	254.336	1.20313	56.5879	77.3333
GTATATATATAA	67	21.6862	67	23.252	2.88147	70.9062	75.5772
TTTTTTTTTTAA	515	448.23	524	482.674	1.08562	43.0464	71.5132
CTGTTTTTTTTT	385	322.843	389	347.212	1.12035	44.2072	67.7894
TTTTTTTCTTTC	358	297.952	358	320.362	1.11749	39.7674	65.7287
GTTTTTTTTTTG	317	260.131	318	279.589	1.13738	40.9366	62.6761
TTTTTTCTTTCT	365	310.849	366	334.272	1.09492	33.1877	58.6149
TCTTTTTTTTTT	773	718.761	785	776.107	1.01146	8.944	56.2355
TTTTTTGTTTTT	448	395.351	461	425.504	1.08342	36.9367	56.009
CATATATATATT	67	30.0885	67	32.2638	2.07663	48.9601	53.6369
AGTTTTGATTTT	169	124.04	180	133.134	1.35202	54.2882	52.2715
TTTTTTTTTGTG	322	274.47	322	295.044	1.09136	28.152	51.4261
GATTTTTTTTTT	421	372.994	424	401.352	1.05643	23.2757	50.9709
AATATATATATT	68	32.7741	68	35.1444	1.93487	44.8828	49.6312
TTTTTTTTTTCT	774	726.676	782	784.715	0.99654	-2.7101	48.833
TTTTATTTTTTT	280	236.136	283	253.737	1.11533	30.8889	47.7071
TTTTTGTTTGTT	340	295.708	342	317.941	1.07567	24.947	47.4552
TTTTTGTTTTTT	434	390.031	450	419.756	1.07205	31.308	46.3592
TTTTTTTTTTGG	501	457.057	504	492.223	1.02393	11.9168	45.9913

Table A.140: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GTTTTTTTTTTGT	201	138.548	202	149.709	1.34929	60.5143	74.7902
CTTCTTCTTTTTT	174	125.188	175	135.255	1.29385	45.0843	57.2868
GTTTTTTTTTTTC	118	73.2873	118	79.1383	1.49106	47.1395	56.2032
GTTTTTTTTTTTG	185	139.893	185	151.164	1.22383	37.3678	51.7037
TTTCTTCTTCTTT	131	89.1984	131	96.3355	1.35983	40.2643	50.3477
TTTTTTGTTTTTT	223	178.021	231	192.44	1.20038	42.1886	50.2352
TTTTTTTTCTTTT	315	268.646	317	290.673	1.09057	27.4843	50.1411
TTTTTTTTTTTGA	222	178.62	222	193.089	1.14973	30.9753	48.2662
TTTTTTTTTCTTT	386	341.296	389	369.553	1.05262	19.9495	47.512
ATTTTTTTTTTTA	89	53.7063	89	57.9824	1.53495	38.1362	44.9545
TTTTTGTTTTTTT	264	223.246	271	241.439	1.12243	31.3005	44.2656
TTTTTGTTTGTTT	223	184.154	224	199.082	1.12516	26.416	42.6817
TTTTTGTTTTGTT	246	207.216	247	224.066	1.10236	24.07	42.2067
TTTTTTTTTTTGC	221	182.699	222	197.506	1.12402	25.9541	42.0616
TTTTTTTTTTTAA	343	303.495	349	328.496	1.06242	21.1306	41.9714
TAAAAAAAAAAAT	56	26.8478	56	28.9775	1.93253	36.8946	41.1694
TTTTTTTTTTTGG	332	293.852	333	318.028	1.04708	15.3187	40.523
ATGTTTTTTTTTT	235	198.636	235	214.77	1.09419	21.1541	39.5056
TCTTTTTTTTTTT	473	435.647	482	472.169	1.02082	9.93217	38.9108
TTTTTTTTGTTTT	327	290.646	337	314.547	1.07138	23.2356	38.5388
TTGTTTTTTTTTG	112	79.3945	114	85.7385	1.32962	32.4782	38.5358
AGTTTTTTTTTTT	359	322.579	361	349.22	1.03373	11.9764	38.4042
TTTTTTGTTGTTG	145	111.496	146	120.444	1.21218	28.093	38.0982
TGTTTTTTTTTTG	171	137.388	172	148.454	1.1586	25.3211	37.4235
CTCTTTTTTTTTT	216	183.145	216	197.99	1.09097	18.8059	35.6395

Table A.141: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTTGG	248	205.173	248	223.326	1.11048	25.989	47.0149
TTTTTTTTTTTTTG	650	610.076	655	666.818	0.982277	-11.7125	41.2026
AGTTTTTTTTTTTT	246	208.648	246	227.117	1.08314	19.647	40.5121
GTTTTTTTTTTTGT	104	71.4341	104	77.6475	1.33939	30.39	39.0641
TTTTTTTTTTCTTT	241	206.028	243	224.26	1.08357	19.5025	37.7848
CTTTTTTTTTTTTC	65	36.6507	65	39.8244	1.63217	31.844	37.242
TGTTTTTTTTTTTG	107	75.7407	107	82.3324	1.29961	28.0408	36.9698
TTTTTTTTTGTTTT	229	195.883	239	213.194	1.12104	27.308	35.7707
TTTTTTTTTTTCTG	80	51.8916	80	56.3939	1.41859	27.9733	34.6296
TTTTTTTTCTTTTT	159	128.834	159	140.123	1.13472	20.0948	33.4501
TTTTTTTTTCTTTT	212	181.597	213	197.617	1.07784	15.9668	32.8165
CTTTTTTTTTTCTT	99	71.9862	99	78.2481	1.26521	23.2883	31.5459
TTTTTGTTTGTTTG	113	85.6359	113	93.0982	1.21377	21.8918	31.333
TTTTTTTTTTTTGA	165	136.878	165	148.884	1.10824	16.9581	30.8311
TTTTTTTTTTTTAA	229	201.477	231	219.295	1.05338	12.0119	29.3229
TCTTTTTTTTTTTT	324	296.306	328	322.826	1.01603	5.21566	28.9493
TTTTTGTTTTTTTT	152	127.118	157	138.254	1.13559	19.963	27.1724
GTTTTTTTTTTTTG	105	82.0341	105	89.1793	1.1774	17.1477	25.9167
CTCTTTTTTTTTTT	145	121.95	145	132.627	1.09329	12.9333	25.1024
GATTTTTTTTTTTT	166	143.347	166	155.931	1.06458	10.3878	24.356
GTTTTTTTTTTCTT	113	91.3298	113	99.2941	1.13803	14.6111	24.059
ATTTTTTTTTTTTT	344	320.933	346	349.745	0.989292	-3.72511	23.8763
TTTTTTTTTTCCTT	107	86.8242	108	94.3913	1.14417	14.5457	22.3569
TTTTTTTCTTTTTT	124	104.03	125	113.117	1.10506	12.4869	21.7749
TTTTTTCTTTTTTT	109	89.3306	110	97.1186	1.13264	13.7002	21.6914

Table A.142: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTTTTG	487	449.114	490	493.365	0.993179	-3.35381	39.441
AGTTTTTTTTTTTTT	183	157.293	183	172.27	1.06229	11.0576	27.702
GTTTTTTTTTTTTTT	365	338.422	365	371.342	0.982921	-6.28767	27.5951
TTTTTGTTTTTTTTT	100	77.7362	104	85.0677	1.22256	20.8981	25.185
TTTTTTTTTTTTTGG	188	165.151	188	180.891	1.0393	7.24703	24.3615
TTTTTTTTTTTTTTC	228	206.219	229	225.969	1.01341	3.05139	22.8932
TTTTTTTTTTTTGTG	76	56.6624	76	61.9929	1.22595	15.4822	22.3153
CAAAAAAAAAAAAAA	135	114.842	135	125.721	1.0738	9.6128	21.832
CTTTTTTTTTTTTTT	336	315.159	339	345.732	0.980527	-6.66635	21.5158
GTTTTTTTTTTGTT	63	45.8425	63	50.1495	1.25624	14.372	20.0291
GTTTTTTTTTTGTTT	69	51.6943	70	56.5545	1.23775	14.9304	19.9244
ACTTTTTTTTTTTTT	119	100.797	119	110.33	1.07858	9.00235	19.756
GATTTTTTTTTTTT	129	111.396	129	121.945	1.05786	7.25547	18.927
TTTTTTTTCTTTTT	115	97.6082	115	106.836	1.07642	8.46832	18.8567
TTTTTTGTTTTTTT	96	78.9194	98	86.3636	1.13474	12.3873	18.8084
TTTTTTTTTTTTAA	165	147.485	167	161.512	1.03398	5.58	18.5157
CTCTTTTTTTTTTT	106	90.1565	106	98.6722	1.07426	7.5934	17.1606
TTTTTTTTTTGGTT	98	82.3608	98	90.1328	1.08728	8.20095	17.0381
GATATATATATGTAC	13	3.68094	13	4.025	3.22981	15.2415	16.4032
TTTTTTTTTTAATT	82	67.2503	82	73.5848	1.11436	8.87898	16.2605
TTTTTTTCTTTTTT	89	74.4373	89	81.4549	1.09263	7.88421	15.9025
TTTTGTTTTTGTTTT	84	69.9202	87	76.5084	1.13713	11.1802	15.4108
ATGTTTTTTTTTTT	84	70.4021	84	77.0361	1.0904	7.26953	14.8338
TTCTCTCTCTCTCTT	23	12.1202	23	13.2543	1.73529	12.677	14.7343
TTTTTTCTTTTTTT	79	65.7933	80	71.9895	1.11127	8.44048	14.4515

Table A.143: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTTTTG	387	336.738	388	371.98	1.04307	16.3603	53.8397
GTTTTTTTTTTTTTT	279	251.059	279	277.087	1.0069	1.91958	29.4406
CTTTTTTTTTTTTTT	261	233.219	264	257.349	1.02584	6.7359	29.3733
AGTTTTTTTTTTTTT	144	122.925	144	135.487	1.06283	8.77521	22.7868
TCTTTTTTTTTTTTT	170	150.056	171	165.438	1.03362	5.65456	21.2141
GTTTTTTTTTTGTTT	44	27.6578	44	30.4539	1.44481	16.191	20.4284
CAAAAAAAAAAAAAA	111	93.3714	111	102.882	1.07891	8.43069	19.1969
TATATATATATATATT	138	120.971	139	133.331	1.04252	5.78787	18.1748
TTTTTTTTTTTTTTC	174	157.668	175	173.844	1.00665	1.16021	17.1503
ATTTTTTTTTTTTTT	189	172.788	190	190.545	0.997141	-0.543987	16.9502
GTTTTTTTTTTTTTC	32	19.1469	32	21.0807	1.51798	13.3561	16.435
CTTTTTTTTTTTTTA	31	18.3268	31	20.1776	1.53636	13.3119	16.2943
TTTTTTTTTTTTTGA	97	82.1613	97	90.5191	1.0716	6.70758	16.1045
TTTTTTTTTTTTTAA	119	103.968	119	114.57	1.03867	4.5148	16.0702
TTTTGTTTTTTTTTG	29	17.435	29	19.1955	1.51077	11.966	14.7557
TTTTTTTTTTTTGTG	61	48.408	61	53.3134	1.14418	8.21582	14.1037
TTTTTTTTTTTTTGG	141	127.595	141	140.641	1.00255	0.359236	14.0858
TTATATATATATATAT	102	89.3682	103	98.4665	1.04604	4.63629	13.4852
TTTTTTTTTTTTGTA	49	37.5842	49	41.3881	1.18392	8.27257	12.9966
GTTTTTTTTTCTTTC	27	16.7753	27	18.4691	1.4619	10.2529	12.85
TTTTTTTTTTTTTGC	87	75.3765	87	83.0382	1.04771	4.05488	12.4769
CATATATATATATATA	106	94.3184	106	103.926	1.01996	2.09448	12.3768
TTTCTTCTTCTTCTTT	21	12.1902	21	13.4203	1.56479	9.40276	11.4218
GTTTTTTTTTTTGTT	27	17.8348	27	19.6358	1.37504	8.599	11.1965
CTCTTTTTTTTTTTT	76	65.7177	76	72.3902	1.04987	3.6983	11.0478

Table A.144: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTTTTTTG	292	264.115	293	293.514	0.99825	-0.513172	29.3083
TTTTTTTTTTTTTTTTC	140	119.306	140	132.384	1.05753	7.83128	22.3935
CTTTTTTTTTTTTTTTT	199	179.866	200	199.71	1.00145	0.289911	20.1171
GTTTTTTTTTTTTTTTT	208	190.066	208	211.057	0.985514	-3.03522	18.7552
TTTTTTTTTTTTTTTGT	134	117.748	134	130.653	1.02562	3.38945	17.3253
GATTTTTTTTTTTTTTT	79	63.4812	79	70.3984	1.12218	9.1069	17.2776
ATATATATATATATATT	129	113.997	130	126.486	1.02778	3.56246	15.9497
GTTTTTTTTTTTTTTTA	26	14.2166	26	15.7575	1.65001	13.0203	15.6959
TTTTTTTTTTTTTTGGT	75	61.5452	75	68.25	1.0989	7.0733	14.8288
ATTTTTTTTTTTTTTTT	143	129.518	144	143.731	1.00187	0.269355	14.1606
TTTTTTTTTTTTTTTAA	90	77.2864	90	85.7203	1.04993	4.38476	13.7063
CAAAAAAAAAAAAAAAA	88	76.0559	88	84.3545	1.04322	3.72319	12.8364
TCTTTTTTTTTTTTTTT	132	120.012	133	133.168	0.998737	-0.168035	12.5681
AGTTTTTTTTTTTTTTT	111	99.2184	111	110.071	1.00844	0.932675	12.4549
TTTTTTTTTTTTTTTGC	73	62.1217	73	68.8898	1.05966	4.23045	11.7795
GTCTCTCTCTCTCTCTT	10	3.17146	10	3.51479	2.84512	10.456	11.4839
CCTTTTTTTTTTTTTTT	42	32.3138	42	35.823	1.17243	6.68133	11.0114
TTCTCTCTCTCTCTCTT	14	6.56921	14	7.28064	1.92291	9.15374	10.5933
TATATATATATATATTT	72	63.0864	72	69.9603	1.02916	2.06919	9.5156
TATATATATATATATGT	67	58.1506	68	64.4833	1.05454	3.61086	9.49096
AAAAAAAAAAAAAAACT	41	32.5703	41	36.1075	1.1355	5.2099	9.43701
GTTTTTTTTTTTTGTT	16	8.8844	16	9.84681	1.62489	7.76706	9.41266
ATTTTTTTTTTTTTTG	30	22.0948	30	24.4917	1.22491	6.0859	9.17559
TTTTTTTTTTTTAATT	39	30.9835	39	34.3478	1.13544	4.95389	8.97415
ATTTTTGTTTTTTTTT	19	11.8678	19	13.1538	1.44444	6.98677	8.94152

Table A.145: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTTTTTTGT	109	90.6041	109	101.191	1.07717	8.10315	20.1485
TATATATATATATATATT	121	102.932	122	114.975	1.0611	7.23583	19.5679
GTTTTTTTTTTTTTTTTT	158	141.015	158	157.576	1.00269	0.424813	17.9697
AATATATATATATATATA	104	89.663	107	100.139	1.06852	7.09117	15.4265
TTTTTTTTTTTTTTTTTG	213	198.519	214	221.97	0.964096	-7.82488	14.9964
CATATATATATATATATA	89	75.5171	89	84.3273	1.05541	4.79981	14.6207
AGTTTTTTTTTTTTTTTT	86	74.1081	86	82.7527	1.03924	3.31021	12.7988
TTTTTTTTTTTTTTTGTG	43	32.7422	43	36.5455	1.17662	6.99366	11.719
TTTTTTTTTTTTTTTTTC	105	94.9601	105	106.061	0.99	-1.05529	10.5529
ATTTTTGTTTTTTTTTTG	16	8.3642	16	9.33333	1.71429	8.62394	10.3781
TTCTCTCTCTCTCTCTCA	7	1.69218	7	1.88811	3.70741	9.17233	9.93926
CTCTCTCTCTCTCTCTTT	51	42.4392	51	47.3737	1.07655	3.76164	9.37138
TTTTTTTTTTTTTTTTGC	58	49.3806	58	55.1263	1.05213	2.94735	9.33133
TGTTTTTTTTTTTTTTTT	70	61.4312	70	68.5878	1.02059	1.42663	9.14042
ACTTTTTTTTTTTTTTTT	51	42.7558	51	47.7273	1.06857	3.38245	8.99237
TATATATATATATATATG	75	66.5359	77	74.2913	1.03646	2.75753	8.98096
CAAAAAAAAAAAAAAAG	12	5.80836	12	6.48118	1.85152	7.39205	8.70731
TTCTTTTTTTTTTTTTTT	42	34.1444	42	38.1111	1.10204	4.08088	8.69695
CTTTTTTTTTTTTTTTTT	144	135.599	144	151.515	0.9504	-7.32562	8.65636
CTTTTTTTTTTTTTTTTA	16	9.36692	16	10.4523	1.53076	6.81219	8.56647
ATTTTTTTTTTTTTTTTA	13	6.74437	13	7.52569	1.72742	7.10615	8.53114
TCTTTTTTTTTTTTTTTT	98	90.2167	99	100.758	0.982556	-1.74216	8.10984
CTTTTTTCTTTTCTTTTT	8	2.98738	8	3.33333	2.4	7.00375	7.88035
TTATATATATATATATAT	76	68.5636	76	76.5569	0.992726	-0.554864	7.82588
TATATATATATATATATC	46	38.8241	47	43.3366	1.08453	3.81409	7.80164

Table A.146: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTTTTT	117	98.2792	117	110.525	1.05858	6.66084	20.4003
GTCTCTCTCTCTCTCTCTT	10	1.52456	10	1.71274	5.83859	17.6449	18.8088
TTTTTTTTTTTTTTTTTTG	163	145.979	164	164.253	0.998461	-0.252559	17.9766
ATATATATATATATATATT	113	99.6914	114	112.115	1.01681	1.90076	14.1599
TTTTTTTTTTTTTTTTTTC	83	71.6824	83	80.5913	1.02989	2.44434	12.1674
TTATATATATATATATATA	69	58.9139	69	66.2269	1.04187	2.8304	10.904
CTTTTTTTTTTTTTTTTGA	11	4.34721	11	4.88396	2.25227	8.93133	10.212
TTTTTTTTTTTTTTTTTGT	80	70.811	80	79.6109	1.00489	0.390028	9.76093
GTTTTTTTTTTTTTTTTTT	117	107.823	117	121.271	0.964784	-4.19463	9.55676
AGTTTTTTTTTTTTTTTT	67	58.1139	67	65.3269	1.02561	1.69432	9.53332
CATATATATATATATATAT	82	73.244	82	82.3484	0.995769	-0.347683	9.25963
TTTTTTTTTTTTTTTTGC	46	37.6927	46	42.3618	1.08588	3.79017	9.16209
TCTTTTTTTTTTTTTTTT	72	63.406	72	71.28	1.0101	0.723624	9.15171
GTTTTTTTTTTTTTTTTA	13	6.68512	13	7.51072	1.73086	7.13203	8.64585
ATATATATATATATATATG	71	62.9448	73	70.7611	1.03164	2.27394	8.54998
TATATATATATATATATAA	53	45.2749	53	50.8874	1.04152	2.15589	8.34959
TTTTTTTTTTTTTTTTGG	55	47.4358	55	53.3174	1.03156	1.70887	8.13757
ATCTCTCTCTCTCTCTCTG	4	0.566187	4	0.636069	6.28862	7.35497	7.8205
TTTTTTTTTTTTTTTGTG	38	31.1246	38	34.9776	1.08641	3.14936	7.58435
CTCTTTTTTTTTTTTTTT	38	31.1311	38	34.985	1.08618	3.14138	7.57637
ATTTTTTTTTTTTTTTTG	17	10.9312	17	12.2817	1.38417	5.5267	7.50709
CTCTTCTTCTTCTTCTTCC	4	0.631708	4	0.709677	5.63636	6.91696	7.38249
CCAAAAAAAAAAAAAAAA	21	14.8063	21	16.6364	1.2623	4.89156	7.3388
TTTTTTTTTTTTTTTTCT	50	43.3741	50	48.75	1.02564	1.26589	7.10798
TTCTTTTTTTTTTTTTTT	34	27.8203	34	31.2632	1.08754	2.85329	6.82023

Table A.147: Top 25 Words by $S * ln(\frac{S}{E_s})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CTTTTTTTTTTTTTTTTTT	93	79.225	93	89.6941	1.03686	3.36608	14.9087
TTTTTTTTTTTTTTTTTGT	66	54.1671	66	61.3084	1.07652	4.86669	13.0403
TATATATATATATATATATT	100	88.5993	101	100.317	1.0068	0.684911	12.1046
TTTTTTTTTTTTTTTTTTG	122	111.012	123	125.725	0.978325	-2.69533	11.5145
TCTTTTTTTTTTTTTTTTT	62	51.6872	62	58.5	1.05983	3.60267	11.2793
AATATATATATATATATATA	82	72.2183	84	81.7554	1.02746	2.27513	10.4161
CATATATATATATATATATA	72	63.0052	72	71.3185	1.00956	0.684702	9.60826
GTTTTTTTTTTTTTTTTTA	11	4.84325	11	5.47885	2.00772	7.667	9.02339
TTTTTTTTTTTTTTTTTCT	43	34.9273	43	39.5238	1.08795	3.62476	8.94105
ATTTTTGTTTTTTTTTTGTG	13	6.93033	13	7.84	1.65816	6.57424	8.17754
AGTTTTTTTTTTTTTTTTT	51	43.8397	51	49.6139	1.02794	1.40526	7.71554
TATATATATATATATATATG	64	56.7542	65	64.2383	1.01186	0.766159	7.68983
GAAAAAAAAAAAAAAAAAG	4	0.682889	4	0.772472	5.17818	6.57782	7.07087
ATATATATATATATATATTT	57	50.3626	57	57	1	1.64535e-13	7.05669
TATATATATATATATATATC	39	32.6616	40	36.959	1.08228	3.16276	6.91709
CTCTCTCTCTCTCTCTCTAT	21	15.3208	21	17.3333	1.21154	4.02971	6.62158
TTTTTTTTTTTTTTTTTAAT	20	14.3705	20	16.2581	1.23016	4.14286	6.6111
AAATATATATATATATATAT	41	34.9352	41	39.5327	1.03712	1.49419	6.5632
TTATATATATATATATATAG	5	1.35388	5	1.53149	3.26479	5.91597	6.53233
GTTTTTTTTTTTTTTTTTG	18	12.8101	18	14.4924	1.24203	3.90139	6.1225
AAAAAAAAAAAAAAAAAAC	26	20.6312	26	23.3427	1.11384	2.80312	6.01359
AATATATATATATATATAGG	3	0.411178	3	0.465116	6.45	5.59224	5.96202
TAGTTTTTTTTTTTTTTTT	21	15.838	21	17.9186	1.17197	3.33234	5.92432
TTTTTTTTTTTTTTTTGGT	30	24.6375	30	27.8767	1.07617	2.20217	5.90786
GAGTCGAATATGACTTGATG	45	39.7645	45	45	1	1.39888e-13	5.56591

A.8 The *Arabidopsis thaliana* Full Genome

A.8.1 Timing Information

Table A.148: Full Size and Timing Information, Full Genome

Length	Order	Size (GB)	Time (s)			
			Enumerate	Score	Cluster	Total
1	0	0.000000	13.359404	16.600909	208.636442	29.960313
2	0	0.000001	24.911273	15.460624	370.131268	40.371897
3	1	0.000003	39.911656	12.949826	192.810054	52.861482
4	2	0.000012	58.588294	11.844762	75.959112	70.433056
5	3	0.000046	79.287054	11.935195	32.825013	91.222249
6	4	0.000186	101.382673	13.816576	14.396031	115.199249
7	5	0.000743	131.089892	12.124423	4.621820	143.214315
8	6	0.002971	156.262442	15.028098	0.471711	171.29054
9	7	0.011884	212.090626	16.626474	1.865925	228.7171
10	8	0.047517	274.062791	32.666027	7.754121	306.728818
11	9	0.188131	337.920171	94.511875	32.235022	432.432046
12	10	0.694700	416.956463	287.657633	117.457379	704.614096
13	11	2.142429	498.078805	766.047459	321.832765	1264.126264
14	12	5.172246	585.895706	1424.738818	625.363307	2010.634524
15	13	9.889705	672.329848	1836.332010	920.130061	2508.661858
16	14	15.799248	761.940257	2145.022116	1161.314575	2906.962373
17	15	22.334908	847.608003	2818.744812	1357.162634	3666.352815
18	16	8.927683	151.350	2013.160	2704.420	4868.93
19	17	11.144783	175.240	2111.470	2584.820	4871.53
20	19	13.386126	179.360	2260.60	2670.260	5110.22

A.8.2 Top 25 Words, $O * ln(\frac{O}{E})$

Table A.149: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 1

Word	S	E_s	O	E	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
C	5	5	21440282	2.14403e+07	1	0	0
G	5	5	21419471	2.14195e+07	1	0	0
T	5	5	38027796	3.80278e+07	1	0	0
A	5	5	38072592	3.80726e+07	1	0	0

Table A.150: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 2

Word	S	E_s	O	E	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
CG	5	5	2783857	3.86045e+06	0.721123	-910171	0
CC	5	5	4031850	3.8642e+06	1.04339	171236	0
GC	5	5	3565695	3.86045e+06	0.923648	-283203	0
CA	5	5	7556040	6.86185e+06	1.10117	728173	0
GA	5	5	7622269	6.85519e+06	1.1119	808476	0
AT	5	5	10998968	1.21706e+07	0.903732	-1.11334e+06	0
GG	5	5	4014076	3.8567e+06	1.04081	160543	0
AC	5	5	6228852	6.86185e+06	0.907751	-602863	0
AG	5	5	7069045	6.85519e+06	1.0312	217153	0
CT	5	5	7068460	6.85378e+06	1.03132	218007	0
GT	5	5	6217386	6.84713e+06	0.908028	-599851	0
AA	5	5	13775613	1.21849e+07	1.13054	1.69025e+06	0
TC	5	5	7613817	6.85378e+06	1.11089	800701	0
TG	5	5	7552440	6.84713e+06	1.10301	740452	0
TA	5	5	9118553	1.21706e+07	0.749228	-2.63264e+06	0
TT	5	5	13742861	1.21563e+07	1.13051	1.68588e+06	0

Table A.151: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 3

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATA	5	5	3043268	2.6374e+06	1.15389	435603	0
TAT	5	5	3040034	2.6343e+06	1.15402	435487	0
AAA	5	5	5250463	4.98436e+06	1.05339	273083	0
TTT	5	5	5230503	4.96653e+06	1.05315	270866	0
GAG	5	5	1653882	1.41525e+06	1.16862	257709	0
CTC	5	5	1649718	1.41523e+06	1.16569	252926	0
TCT	5	5	2750549	2.51013e+06	1.09578	251578	0
AGA	5	5	2755246	2.51557e+06	1.09528	250748	0
TGG	5	5	1560151	1.41535e+06	1.10231	151967	0
CCA	5	5	1560238	1.42092e+06	1.09805	145940	0
TGT	5	5	2317847	2.19223e+06	1.0573	129148	0
ACA	5	5	2320505	2.19519e+06	1.05709	128828	0
AAC	5	5	2358886	2.25375e+06	1.04665	107547	0
ACC	5	5	1274289	1.17134e+06	1.08789	107349	0
GTT	5	5	2347645	2.2469e+06	1.04484	102970	0
GGT	5	5	1263418	1.16516e+06	1.08433	102292	0
AGC	5	5	1268561	1.17678e+06	1.07799	95267.9	0
GCT	5	5	1266100	1.17554e+06	1.07703	93958.8	0
TTG	5	5	2819659	2.72938e+06	1.03308	91760.6	0
CAA	5	5	2815806	2.73396e+06	1.02994	83055.3	0
CTT	5	5	2622044	2.55447e+06	1.02645	68460	0
AAG	5	5	2623823	2.55776e+06	1.02583	66912.3	0
CCG	5	5	584837	523505	1.11716	64792.2	0
CAT	5	5	2228528	2.1829e+06	1.0209	46102.1	0
GTG	5	5	1278900	1.23479e+06	1.03572	44886.1	0

Table A.152: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 4

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAAG	5	5	661010	510311	1.29531	171036	0
CTTC	5	5	660112	510580	1.29287	169558	0
TATA	5	5	958723	841137	1.13979	125447	0
ATAT	5	5	1132846	1.01459e+06	1.11655	124889	0
ATTA	5	5	881531	777615	1.13363	110569	0
TAAT	5	5	882846	779179	1.13305	110276	0
TGAT	5	5	796263	707273	1.12582	94367.1	0
ATCA	5	5	794109	707217	1.12287	92024.1	0
TAGT	5	5	472878	404652	1.1686	73679.2	0
ACTA	5	5	473148	406571	1.16375	71752.8	0
CTGC	5	5	255401	197111	1.29572	66166.5	0
CATC	5	5	505323	443511	1.13937	65931.9	0
GCAG	5	5	256401	198323	1.29285	65856	0
GATG	5	5	503841	442430	1.1388	65488.5	0
AGAA	5	5	1030683	968472	1.06424	64167.2	0
TTCT	5	5	1028162	966758	1.06352	63314.3	0
AAAC	5	5	956299	899070	1.06365	59012.7	0
GTTT	5	5	944465	893509	1.05703	52382.6	0
ACAA	5	5	908301	864751	1.05036	44628.8	0
TGCA	5	5	454548	412609	1.10164	44001.6	0
TTGT	5	5	907911	865355	1.04918	43586.2	0
GACG	5	5	179961	141416	1.27257	43377.1	0
CGTC	5	5	179561	141506	1.26893	42766.2	0
TCTT	5	5	1061349	1.02032e+06	1.04022	41847.6	0
GGAG	5	5	349635	311105	1.12385	40823.4	0

Table A.153: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 5

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATAT	5	5	405703	356881	1.1368	52018.5	0
ATATA	5	5	405440	357261	1.13486	51290.7	0
TTCTT	5	5	437210	396735	1.10202	42472.9	0
AAGAA	5	5	437996	397525	1.10181	42464.2	0
AATAA	5	5	357421	324122	1.10274	34953.6	0
TTATT	5	5	355319	323298	1.09904	33556.6	0
GAAGA	5	5	297369	267716	1.11076	31238.3	0
TCTTC	5	5	296826	267200	1.11088	31211.3	0
AATTA	5	5	310379	280997	1.10456	30867.2	0
TAATT	5	5	310424	281435	1.103	30433	0
GATTC	5	5	156352	129578	1.20663	29367.5	0
GAATC	5	5	155493	129571	1.20006	28357.2	0
TATTT	5	5	404970	378279	1.07056	27611.7	0
AAAAA	5	5	807292	780750	1.034	26988.6	0
AGAGA	5	5	280853	255151	1.10073	26954.7	0
AAATA	5	5	404756	378974	1.06803	26639.4	0
TCTCT	5	5	279302	254329	1.09819	26160.8	0
GATGA	5	5	193850	170237	1.13871	25180.3	0
TCATC	5	5	194045	170532	1.13788	25064.8	0
TTTTT	5	5	803582	780031	1.03019	23902.8	0
TGAAT	5	5	220379	197902	1.11358	23707.7	0
ATTCA	5	5	220107	197932	1.11203	23373.3	0
TTGTT	5	5	375314	355653	1.05528	20194.5	0
AACAA	5	5	375483	355865	1.05513	20149.1	0
ACATA	5	5	192156	173073	1.11026	20098.8	0

Table A.154: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 6

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAA	5	5	376532	321889	1.16976	59038.4	0
TTTTTT	5	5	372939	319693	1.16655	57453	0
TATATA	5	5	179209	145199	1.23423	37714	0
ATATAT	5	5	193956	171570	1.13048	23786.6	0
TAAAAT	5	5	128179	111987	1.14458	17309.4	0
CAAAAC	5	5	89769	74581.8	1.20363	16638	0
ATTTTA	5	5	127290	112000	1.13652	16289.2	0
GTTTTG	5	5	89452	74620.1	1.19877	16216.9	0
AAAATA	5	5	170561	156185	1.09205	15018.3	0
TATTTT	5	5	170110	156432	1.08744	14259.5	0
AATATT	5	5	118527	105726	1.12107	13546	0
TTTTGT	5	5	151587	139086	1.08988	13046.5	0
ACAAAA	5	5	151800	140221	1.08258	12044.8	0
TAATTA	5	5	96388	85321.2	1.12971	11755.4	0
AGAGAG	5	5	79288	68595.2	1.15588	11486	0
CTCTCT	5	5	78300	67897.2	1.15321	11161.9	0
TTAAAA	5	5	146664	137268	1.06845	9710.4	0
CTAAAC	5	5	46509	37911.3	1.22678	9506.23	0
AAAACA	5	5	146007	137146	1.06461	9140.94	0
TTATAA	5	5	92002	83442.4	1.10258	8984.33	0
AGAAGA	5	5	129988	121358	1.07111	8929.33	0
TGAAAT	5	5	81607	73154	1.11555	8923.61	0
TTTTAA	5	5	146121	137507	1.06264	8878.32	0
TATTAT	5	5	92345	83960.6	1.09986	8789.75	0
ATTTCA	5	5	81199	72979.2	1.11263	8666.25	0

Table A.155: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 7

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAAG	5	5	47564	57235.9	0.831016	-8804.38	0
AAAAAAC	5	5	44777	57431.8	0.779655	-11145.2	0
AAAAACA	5	5	53756	50043.5	1.07419	3846.94	0
AAAAAGC	5	5	16325	19929.8	0.819123	-3257.18	0
AAAAAAT	5	5	72222	86242.2	0.837432	-12813.3	0
AAAAACG	5	5	14135	14380.3	0.982942	-243.195	0
AAAAACC	5	5	20781	24789.9	0.838286	-3665.69	0
AAAAACT	5	5	34463	33921	1.01598	546.307	0
AAAAAGA	5	5	53473	49991.3	1.06965	3600.18	0
AAAAATC	5	5	33409	35808.3	0.932995	-2317.09	0
AAAAATG	5	5	35803	35293.1	1.01445	513.565	0
AAAACAC	5	5	21672	24226.4	0.89456	-2414.78	0
AAAACAG	5	5	22628	22199.5	1.0193	432.598	0
AAAAATT	5	5	49433	53653.6	0.921336	-4050.07	0
AAAACAA	5	5	62627	60718.1	1.03144	1938.6	0
AAAAAGG	5	5	21740	21443	1.01385	299.016	0
AAAACCG	5	5	9394	10454.4	0.898573	-1004.66	0
AAAAAGT	5	5	31177	31350.8	0.994457	-173.304	0
AAAACGC	5	5	6330	7157.52	0.884385	-777.722	0
AAAACGA	5	5	16351	15853.5	1.03138	505.217	0
AAAACCT	5	5	19578	18421.9	1.06276	1191.67	0
AAAACGG	5	5	6623	7307.28	0.906357	-651.188	0
AAAACGT	5	5	12652	11637.7	1.08716	1057.27	0
AAAACCC	5	5	15606	16522.6	0.944527	-890.654	0
AAAAAAA	5	5	211967	175620	1.20697	39873.1	0

Table A.156: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 8

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAAAA	5	5	128647	119326	1.07811	9676.06	0
TTTTTTTT	5	5	126568	117334	1.0787	9588.35	0
TATATATA	5	5	58214	49387.5	1.17872	9572.01	0
ATATATAT	5	5	59429	53451.6	1.11183	6299.82	0
TAAAAAAT	5	5	14818	11271.2	1.31467	4054.04	0
ATTTTTTA	5	5	14746	11383.8	1.29535	3816.04	0
GAAGAAGA	5	5	30103	26908.1	1.11873	3377.47	0
TCTTCTTC	5	5	30266	27088.9	1.11728	3356.55	0
TTTTAAAA	5	5	29355	26314.5	1.11555	3209.8	0
AATATATT	5	5	14169	11351.2	1.24823	3141.69	0
TTTTCTTT	5	5	31061	28169	1.10267	3035.64	0
AAAGAAAA	5	5	31032	28186.8	1.10094	2984.2	0
AGAGAGAG	5	5	19380	16632	1.16522	2963.45	0
TCTCTCTC	5	5	19178	16519.1	1.16096	2862.28	0
GAGAGAGA	5	5	20068	17415.3	1.15232	2845.22	0
AAGAAGAA	5	5	32394	29730.1	1.0896	2779.82	0
CTCTCTCT	5	5	18513	15955.2	1.16031	2752.62	0
AGAAGAAG	5	5	26477	24049.6	1.10093	2545.95	0
TTATATAA	5	5	11402	9137.95	1.24776	2523.87	0
TTCTTCTT	5	5	32331	29907	1.08105	2519.71	0
CTTCTTCT	5	5	26462	24182.9	1.09424	2383.28	0
TTTTTCTT	5	5	30556	28327.7	1.07866	2313.73	0
AAGAAAAA	5	5	30460	28234.6	1.07882	2310.86	0
TTTGTTTT	5	5	32138	29929.8	1.07378	2287.78	0
AAATTTTA	5	5	15823	13781.9	1.1481	2185.33	0

Table A.157: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 9

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATATATAT	5	5	40577	35298.1	1.14955	5655.34	0
ATATATATA	5	5	40448	35398.1	1.14266	5394.08	0
TTTTTTTTT	5	5	81519	76580.3	1.06449	5094.59	0
AAAAAAAAA	5	5	82900	78078.4	1.06175	4967.47	0
GAGAGAGAG	5	5	11477	8804.03	1.30361	3042.97	0
CTCTCTCTC	5	5	10740	8290.54	1.29545	2780.16	0
TAAAAAAAT	5	5	6975	4862.13	1.43456	2516.97	0
ATTTTTTTA	5	5	7026	4951.7	1.41891	2458.3	0
AAGAAGAAG	5	5	16422	14159.7	1.15977	2434.09	0
CTTCTTCTT	5	5	16290	14113.9	1.15418	2335.83	0
AATATATAA	5	5	5374	3677.17	1.46145	2039.06	0
AAAATAAAA	5	5	16404	14605.9	1.12311	1904.48	0
TTATATATT	5	5	5215	3638.6	1.43324	1877.08	0
TTTTATTTT	5	5	16374	14628.1	1.11935	1846.18	0
TTTTTCTTT	5	5	14627	12990	1.12602	1736.09	0
AAAGAAAAA	5	5	14565	12977	1.12237	1681.38	0
AGAGAGAGA	5	5	14009	12442.2	1.12593	1661.57	0
AAAAGAAAA	5	5	15021	13468.8	1.11524	1638.35	0
TCTCTCTCT	5	5	13272	11755.2	1.12903	1610.7	0
TTTTCTTTT	5	5	15049	13532.4	1.11207	1598.54	0
TCTTCTTCT	5	5	16939	15585	1.08688	1411.16	0
CCTAAACCC	5	5	3138	2023.78	1.55056	1376.38	0
TTTTGTTTT	5	5	16232	14954.1	1.08545	1331	0
GAAGAAGAA	5	5	16563	15284.6	1.08364	1330.45	0
TTCTTCTTC	5	5	16633	15356.5	1.08313	1328.17	0

Table A.158: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 10

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TATATATATA	5	5	31261	27617.1	1.13194	3874.33	0
ATATATATAT	5	5	31669	28193.5	1.12327	3681.38	0
CTTCTTCTTC	5	5	10149	8380.55	1.21102	1943.14	0
GAAGAAGAAG	5	5	9989	8396.54	1.18966	1734.73	0
ATTTTTTTTA	5	5	3285	2222.47	1.47808	1283.6	0
AATATATATT	5	5	2446	1450.01	1.68688	1278.97	0
TAAAAAAAAT	5	5	3201	2164.42	1.47892	1252.58	0
AGAGAGAGAG	5	5	9154	8011.82	1.14256	1219.98	0
TTTTTTTTTT	5	5	53705	52504.2	1.02287	1214.46	0
CTCTCTCTCT	5	5	8487	7432.54	1.14187	1125.95	0
GAGAGAGAGA	5	5	9339	8296.25	1.12569	1105.7	0
TTATATATAA	5	5	1999	1150.63	1.73731	1104.12	0
AGAAGAAGAA	5	5	10233	9232.54	1.10836	1052.81	0
AAAAAAAAAA	5	5	54460	53420.7	1.01946	1049.37	0
TCTCTCTCTC	5	5	8624	7699.52	1.12007	977.88	0
TTTTTAAAAA	5	5	5302	4411.27	1.20192	975.149	0
TTCTTCTTCT	5	5	10188	9309.01	1.09442	919.245	0
TCTTCTTCTT	5	5	11238	10427.6	1.07771	841.058	0
GATGATGATG	5	5	3542	2821.89	1.25519	805.037	0
AAAGAAGAAA	5	5	3063	2369.44	1.29271	786.394	0
CATCATCATC	5	5	3471	2770.69	1.25276	782.173	0
GTTGTTGTTG	5	5	2666	2037.07	1.30874	717.326	0
AATAAAATAA	5	5	4153	3497.23	1.18751	713.731	0
CAACAACAAC	5	5	2707	2096.48	1.29121	691.863	0
AAGAAGAAGA	5	5	11075	10407.6	1.06413	688.39	0

Table A.159: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 11

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATATATATATA	5	5	26398	24398.2	1.08197	2079.64	0
TATATATATAT	5	5	26436	24476	1.08008	2036.48	0
TTTTTTTTTTT	5	5	36702	35381	1.03734	1345.32	0
AAAAAAAAAAA	5	5	37064	35776.7	1.03598	1310.15	0
CTCTCTCTCTC	5	5	6324	5514.76	1.14674	865.907	0
GAGAGAGAGAG	5	5	6916	6102.45	1.13332	865.52	0
ATTTTTTTTTA	5	5	1820	1135.15	1.60331	859.17	0
GAAGAAGAAGA	5	5	7529	6736.58	1.11763	837.295	0
TCTTCTTCTTC	5	5	7740	7001.5	1.10548	776.144	0
TAAAAAAAAAT	5	5	1734	1121.7	1.54586	755.298	0
CAAGAAGAAGC	5	5	713	266.764	2.67278	700.963	0
TTATATATATT	5	5	950	469.817	2.02207	668.914	0
TAAACCCTAAA	5	5	2503	1951.35	1.2827	623.168	0
AATATATATAA	5	5	903	463.57	1.94792	602.088	0
CTTTTAAAATC	5	5	716	319.735	2.23935	577.23	0
GATTTTAAAAG	5	5	789	389.378	2.02631	557.204	0
TTCTTCTTCTT	5	5	7230	6759.12	1.06967	486.913	0
CTTATAATTAG	5	5	691	344.414	2.0063	481.139	0
AAGAAGAAGAA	5	5	7193	6753.9	1.06501	453.072	0
GCTTCTTCTTG	5	5	510	211.082	2.41612	449.903	5.55112e-15
TAAACACTAAA	5	5	711	391.714	1.8151	423.855	0
GTGTTTTGGAG	5	5	821	497.609	1.64989	411.082	0
GTTGTCTAAAC	5	5	643	346.074	1.85798	398.333	0
AGAAGAAGAAG	5	5	6549	6171.43	1.06118	388.889	0
TTATATATATG	5	5	599	313.922	1.90812	387.024	0

Table A.160: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 12

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTTTTTTTTTTT	5	5	26676	25082.1	1.06355	1643.45	0
AAAAAAAAAAAA	5	5	26775	25224.8	1.06146	1596.94	0
ATATATATATAT	5	5	23461	22323.6	1.05095	1165.91	0
TATATATATATA	5	5	22840	22036	1.03649	818.51	0
TAAACCCTAAAC	5	5	1957	1454.09	1.34586	581.292	0
GTTTTTTTTTTC	5	5	916	505.626	1.81161	544.304	0
GAAAAAAAAAAC	5	5	916	525.385	1.74348	509.191	0
ATTTTTTTTTTA	5	5	920	571.622	1.60945	437.823	0
TAAAAAAAAAAT	5	5	883	561.266	1.57323	400.115	0
TTATATATATAA	5	5	418	162.952	2.56518	393.768	1.07536e-11
AAGAAGAAGAAG	5	5	4946	4603.44	1.07441	355.005	0
GAAAAAAAAAAT	5	5	798	514.733	1.55032	349.891	0
CTTCTTCTTCTT	5	5	4976	4666	1.06644	320.073	0
CTTTTTTTTTTC	5	5	735	479.303	1.53348	314.241	0
ATTTTTTTTTTC	5	5	775	517.021	1.49897	313.704	0
AATATATATATT	5	5	455	230.279	1.97587	309.858	0
GAAAAAAAAAAG	5	5	750	503.993	1.48812	298.133	0
GTTAGTGTTTTG	5	5	884	644.331	1.37196	279.559	0
AGAGAGAGAGAG	5	5	6037	5765.18	1.04715	278.125	0
AAAAAATAAAAA	5	5	1663	1417.67	1.17305	265.424	0
GCAAGAAGAAGC	5	5	501	302.376	1.65688	252.973	0
TTCTTCTTCTTC	5	5	5225	4979.55	1.04929	251.401	0
GAAGATACAAAG	5	5	694	487.775	1.42279	244.716	0
TTTTATTTTTTT	5	5	1602	1377.19	1.16324	242.24	0
CTCTCTCTCTCT	5	5	5486	5250.45	1.04486	240.761	0

Table A.161: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 13

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AAAAAAAAAAAAA	5	5	20022	19342.2	1.03514	691.572	0
TTTTTTTTTTTTT	5	5	19995	19388.8	1.03126	615.545	0
ATATATATATATA	5	5	20692	20269.7	1.02084	426.69	0
GAAGAAGAAGAAG	5	5	3878	3508.2	1.10541	388.638	0
TATATATATATAT	5	5	20682	20298.9	1.01888	386.736	0
CTTCTTCTTCTTC	5	5	3930	3596.07	1.09286	348.973	0
CTCTCTCTCTCTC	5	5	4505	4243.99	1.0615	268.879	0
GAGAGAGAGAGAG	5	5	4938	4712.5	1.04785	230.806	0
TAAAAAAAAAAAT	5	5	456	275.004	1.65816	230.603	0
ATTTTTTTTTTTA	5	5	462	283.046	1.63224	226.36	0
AGAAGAAGAAGAA	5	5	3680	3489.88	1.05448	195.208	0
ATTTATTTTATTA	5	5	249	113.968	2.18482	194.602	2.51261e-08
TTTTATTTTATTT	5	5	1248	1069.97	1.16638	192.078	0
AAAGAAGAAGAAA	5	5	389	238.849	1.62864	189.734	0
TTCTTCTTCTTCT	5	5	3673	3492.78	1.0516	184.787	0
TTTTTTATTTTTT	5	5	832	668.667	1.24427	181.83	0
TTTCTTCTTCTTT	5	5	386	241.241	1.60006	181.436	0
AAATAAAATAAAA	5	5	1194	1032.2	1.15675	173.865	0
AAGAAGAAGAAGA	5	5	4053	3889.43	1.04205	166.959	0
TTATATATATATT	5	5	212	96.8177	2.18968	166.156	3.85108e-07
TAATAAAATAAAT	5	5	243	123.364	1.96979	164.736	5.65359e-09
AAAAAATAAAAAA	5	5	836	694.658	1.20347	154.834	0
CAAGAAGAAGCTT	5	5	506	372.708	1.35763	154.705	0
CAAAACACTAACC	5	5	284	168.837	1.68209	147.691	4.2466e-12
CAACAACAACAAC	5	5	925	789.425	1.17174	146.603	0

Table A.162: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 14

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATCAAAGCTTTGAG	5	5	435	182.073	2.38915	378.859	5.26246e-13
CTCAAAGCTTTGAT	5	5	548	322.807	1.69761	290.013	0
TTTTTTTTTTTTT	5	5	15191	14987.3	1.01359	205.123	0
AAAAAAAAAAAAAA	5	5	15170	14972.2	1.01321	199.107	0
TTTTGAGAAGCAAT	1	4.99724	148	41.9118	3.53123	186.724	-1.60889
ATATATATATATAT	5	5	18917	18737	1.00961	180.912	0
TTAAACCCTAAACA	4	4.98564	106	32.0848	3.30374	126.676	-0.88107
CTTTGAGAAGCAAG	5	5	496	389.305	1.27407	120.138	0
TTCTTCTTCTTCTT	5	5	2962	2845.97	1.04077	118.362	0
AAGAAGAAGAAGAA	5	5	3001	2896.12	1.03621	106.752	0
GTTTTTGGTTTTGA	5	5	213	130.254	1.63526	104.754	1.89587e-09
CAAGCTTCTTCTTG	5	5	305	217.121	1.40475	103.657	2.22045e-15
ATTTTTTTTTTTTA	5	5	244	159.95	1.52548	103.043	1.72751e-11
TTAAACACTAAACA	5	4.99996	140	68.186	2.05321	100.717	3.78602e-05
TTTTTATTTTTTTT	5	5	548	466.373	1.17503	88.3871	0
GTGGTTAGTGTTTA	5	1.57949	31	1.91054	16.2258	86.3846	5.76169
TATGTCATGTGTAG	4	1.61805	31	1.96798	15.7522	85.4664	3.6203
GAATCCTACTTTAA	1	4.57828	57	12.8952	4.42025	84.7131	-1.52132
TTTAAAACCTAAAT	5	5	178	112.644	1.5802	81.4446	3.10122e-08
AGAGAGAGAGAGAG	5	5	4448	4367.57	1.01841	81.1615	0
TATAAAATAAAAAT	5	5	154	93.9089	1.63989	76.1726	6.12497e-07
TAAAAAAAAAAAAT	5	5	219	155.472	1.40861	75.0306	3.50497e-11
CTAAACACTAAACC	5	5	223	159.467	1.39841	74.7799	1.86451e-11
AAAGCTTCTTCTTT	5	4.91783	67	21.9866	3.04731	74.6555	0.0828492
AACATGCAAAATAC	5	4.96131	73	26.3024	2.77541	74.5183	0.0388398

Table A.163: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 15

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TAACCCTAAACCCTT	2	4.98049	136	30.2864	4.49047	204.266	-1.82476
AAACCCTAAACCCTA	5	5	983	870.053	1.12982	119.98	0
GAGAGAGAGAGAGAG	5	5	3807	3698.25	1.0294	110.33	0
CTCTCTCTCTCTCTC	5	5	3433	3343.84	1.02666	90.3374	0
CCCCTAAACCCTAAC	1	4.58096	58	12.9298	4.48577	87.0528	-1.52191
AAAACACTAACAAAA	5	4.99133	88	35.0653	2.5096	80.9709	0.00867981
GTAAACCCTAAACCA	2	4.86044	61	18.9973	3.21099	71.1613	-1.77596
ATTTTTTTTTTTTTA	5	5	159	102.782	1.54696	69.3705	1.48874e-07
ATAAGTATTGTATCA	1	4.98192	76	30.7309	2.47308	68.8152	-1.60582
TTTTTTATTTTTTTT	5	5	362	305.709	1.18413	61.1816	0
CCCTACACCCTAAAT	2	4.99416	80	37.4194	2.13793	60.7871	-1.83024
GAAACACTAACAAAT	2	5	179	127.584	1.403	60.6113	-1.83258
AAAAAAAAAAAAAAA	5	5	11552	11493.8	1.00506	58.3453	0
AAGAGAGAGAGAGAA	5	4.996	80	39.6755	2.01636	56.1034	0.00400578
TCCTACACCCTAAAC	2	4.9996	96	53.7151	1.78721	55.7429	-1.83242
TCCAAAACCCTAAAA	4	4.98911	72	33.7177	2.13538	54.6223	-0.883854
AAATATAGCAAAGAC	2	3.56381	33	6.37168	5.17917	54.2733	-1.15537
AACAAAAAAAAAAAC	5	5	154	108.45	1.42001	54.0022	6.04079e-08
CTAAGTATTGTATCC	1	5	127	83.417	1.52247	53.3825	-1.60944
AAAATCTTTTAAAAA	5	4.93338	58	23.1811	2.50204	53.1921	0.0670664
AAAAAAAATAAAAAA	5	5	331	282.043	1.17358	52.9789	0
TAAATTAATAATGTT	5	4.93123	57	23	2.47826	51.7308	0.069244
ATATATATATATATA	5	5	16769	16718.2	1.00304	50.9201	0
TTCTTCTTCTTCTTC	5	5	2436	2386.45	1.02076	50.0655	0
GAATAAATAAATAAT	5	4.84636	50	18.4592	2.70867	49.8229	0.156054

Table A.164: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 16

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
AACCATCAAAGCTTTT	1	4.85926	85	18.9503	4.48542	127.571	-1.58089
TACATTTTGACCATAT	1	4.38384	60	10.8491	5.53043	102.616	-1.47793
TCTAAACCCTAAACCG	1	3.1605	41	5.0859	8.06151	85.5711	-1.15073
GTTTTCCCGCCAAAAC	5	4.96518	79	26.9113	2.93557	85.0752	0.0349453
ATTCTTCTTGCTTCTT	2	4.98617	85	32.3077	2.63095	82.2244	-1.82704
AGAAGAAGAAGAAGAA	5	5	1890	1810.31	1.04402	81.4152	0
GAAGAAGAAGAAGAAG	5	5	1990	1915.09	1.03911	76.3533	0
CACTCATACGGACTTT	1	4.99083	85	34.7353	2.44707	76.0659	-1.6076
ATTTTCCCGCCAAAAA	5	5	383	314.904	1.21624	74.9792	0
AAACCAAGCTTCTTCA	1	4.90699	66	21.2834	3.10101	74.6941	-1.59066
CCTAAACCCTAAACCC	5	5	674	603.356	1.11708	74.6268	0
CTTCTTCTTCTTCTTC	5	5	1963	1890.74	1.03822	73.6263	0
TTCTTCTTCTTCTTCT	5	5	1848	1776.09	1.04049	73.3435	0
GACTCATACGGACTTC	5	5	283	222.246	1.27337	68.3907	1.11022e-15
CACCATCAAAGCTTTG	5	5	372	311.145	1.19558	66.4516	0
GGTGTCGATCGACACC	5	5	279	220.126	1.26746	66.1268	2.22045e-15
GTCTTCTAACAAGGAT	4	4.3967	45	10.9622	4.10501	63.5494	-0.378241
TGTGTCGATCGACACT	5	4.92696	62	22.6562	2.73656	62.4154	0.0735805
TAACCAAGCTTCTTCT	5	5	236	183.335	1.28726	59.5938	4.31877e-13
CCTCCCCCAAACTTAA	5	4.99928	94	50.0989	1.87629	59.1538	0.000715202
CCACTCAACCAGACAT	5	4.5245	45	12.2449	3.675	58.5699	0.499649
GTTTTGGCGGGAAAAC	5	4.66786	48	14.1962	3.38118	58.4748	0.343684
ACCTAAACCCTAAACG	1	4.18736	40	9.37089	4.26854	58.0509	-1.43207
TTAAGTTTGGGGGAGG	5	4.99574	80	39.2982	2.03571	56.8677	0.00426616
CAAATCCCAAAACCTT	5	4.04794	38	8.53125	4.45421	56.7663	1.05615

Table A.165: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 17

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTCTTCTTGCTTCTCT	1	4.86953	81	19.375	4.18065	115.868	-1.583
CAACACTAAACCTAAAA	1	4.97349	80	28.4942	2.80759	82.5861	-1.60412
GAGAGAGAGAGAGAGAG	5	5	3115	3035.22	1.02628	80.8199	0
TATTTTCCCGCCAAAAC	5	4.16875	47	9.25054	5.08079	76.3969	0.909113
AAACTAGAACCGCAACG	2	4.99056	83	34.563	2.40141	72.7126	-1.8288
TAACCGGATCTTAAAAA	1	4.92086	66	22.2	2.97297	71.9111	-1.59348
GCATCTAGTCATATTTG	2	5	168	113.076	1.48572	66.5112	-1.83258
CTCTCTCTCTCTCTCTC	5	5	2755	2691.08	1.02375	64.6747	0
ATTTTGGAGTCAAATAA	1	2.67291	31	3.87435	8.00135	64.4679	-0.983167
CTTCTTCTTGCTTCTCA	5	5	263	206.569	1.27318	63.5201	1.11022e-14
TAAGAAGAAGCTTGGTA	2	1.887	26	2.38846	10.8857	62.0736	0.116317
AAACACTAAACCTAAAC	2	5	154	107.012	1.4391	56.0584	-1.83258
TTTTTCCCGCCAAAAAA	5	4.19338	39	9.41043	4.14434	55.448	0.879653
TTTAAAAGCCTAAGTAT	1	4.99989	104	61.5553	1.68954	54.5434	-1.60942
GGACATCTAGTCATATA	1	2.24431	24	3.00926	7.97538	49.8326	-0.808396
AAGTATATTTGCTCCTC	1	4.96149	60	26.3288	2.27888	49.4209	-1.60171
ACATCTAGTCATATTTC	2	4.99982	97	58.5852	1.65571	48.9103	-1.83251
AGCTTCTCAAAGCTTTC	3	3.74131	32	7.0614	4.53168	48.3549	-0.662468
TTTTTTGGCGGGAAAAA	5	4.28279	37	10.0364	3.68659	48.274	0.774168
CTTAAAAGCCTAAGTAG	4	5	146	104.905	1.39173	48.2601	-0.892574
CAACCGGATCTTAAAAG	4	5	144	103.096	1.39676	48.1185	-0.892574
TACCAAGATTCTTCTTA	1	3.02432	27	4.71687	5.72414	47.1067	-1.10669
GAACTAGAACCGCAACC	5	4.99989	98	61.7067	1.58816	45.3323	0.000107911
TAGTATATTTGCTCCTA	1	4.99873	79	46.6164	1.69468	41.6721	-1.60918
ATAATTAATTAAATTAT	5	4.44274	35	11.3889	3.07317	39.2948	0.59083

Table A.166: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 18

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
ATTGTTAGAAGATACAAT	1	4.38379	74	10.8486	6.82116	142.082	-1.47791
TAAGAAGCTTGGTTAGTA	1	4.72726	54	15.2768	3.53477	68.183	-1.55335
GAAGAAGCTTGGTTAGTG	4	5	440	381.245	1.15411	63.0665	-0.892574
AGATGTCATGTGTATGAG	1	5	214	163.47	1.30911	57.6402	-1.60944
CTTGTTAGAAGATACAAA	5	5	400	346.673	1.15383	57.2336	0
AGAGAGAGAGAGAGAGAG	5	5	2935	2878.85	1.0195	56.6897	0
AACATCTAGTCATATTTA	1	4.79525	51	16.8605	3.02483	56.4496	-1.56763
CCAAAGCTTTGATGGTGA	1	4.99828	84	44.7813	1.87578	52.8382	-1.60909
TGATGTCATGTGTATGAC	1	4.98139	67	30.5622	2.19225	52.5902	-1.60571
ATATATATATATATATATAT	5	5	12509	12465.8	1.00346	43.2452	0
CCTAACTCTTATAATTAT	1	3.51438	28	6.19523	4.51961	42.2359	-1.25686
TAAGAAGCTTGGTTAGTT	3	4.10701	32	8.8704	3.6075	41.0565	-0.94225
CTTTAAAAGATTTACAAA	5	3.92339	30	7.88235	3.80597	40.0971	1.21241
TCTCTCTCTCTCTCTCTC	5	5	2569	2530.55	1.01519	38.7396	0
TAAACCCTAAAACCTAAC	1	3.44059	26	5.94286	4.375	38.3736	-1.23564
GTAAACCCTAAACCATAA	1	4.97998	58	30.1352	1.92466	37.9754	-1.60543
TATATATATATATATATT	5	5	869	832.371	1.04401	37.423	0
CTCTCTCTCTCTCTCTCT	5	5	2565	2528.13	1.01458	37.1396	0
ATTTTATTAATGTTGATC	5	4.90992	47	21.4646	2.18966	36.836	0.0909062
TCTAACTCTTATAATTAG	2	5	361	326.525	1.10558	36.2342	-1.83258
GATCAACATTAATAAAAT	5	4.90191	46	20.9825	2.19231	36.1079	0.0990671
TCAAAGCTTTGATGGTGT	5	5	217	185.153	1.172	34.441	3.24185e-13
TTTTTATTAATGTTGATT	5	4.99717	68	41.7638	1.62821	33.1485	0.00282938
TGTTACTACATGCAATGT	1	4.99962	81	54	1.5	32.8427	-1.60936
GTTCACTACTCAGACATA	4	4.37295	31	10.7551	2.88235	32.8168	-0.356577

Table A.167: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 19

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
GAATGCATTGGATTGTGAT	1	4.9588	88	25.9389	3.39259	107.5	-1.60116
AAATGCATTGGATTGTGAC	2	5	228	157.31	1.44937	84.617	-1.83258
CAATGCATTGGATTGTGAA	1	4.44362	51	11.3974	4.47471	76.4205	-1.49147
GAGAGAGAGAGAGAGAGAG	5	5	2694	2640.15	1.0204	54.3981	0
TTCTTCTTCTTCTTCTTCT	5	5	1123	1077.28	1.04244	46.677	0
GATCTAGTCATATTTGACA	1	1.67812	20	2.05882	9.71429	45.472	-0.517675
TTAAACCCTAAACCCTAAC	1	4.76086	44	16.0013	2.74977	44.5068	-1.56043
CTCTCTCTCTCTCTCTCTC	5	5	2334	2291.99	1.01833	42.3878	0
TTCATATTTGACTCCAAAG	1	2.67558	23	3.88024	5.92747	40.9307	-0.984166
AGAAGAAGAAGAAGAAGAA	5	5	1137	1097.38	1.03611	40.3312	0
AAAACCTAAACCCTAAACG	1	4.37399	34	10.764	3.15868	39.1052	-1.47568
TAAATCCGCTAACCAAGCC	1	4.93145	49	23.0177	2.1288	37.0223	-1.59563
GGTTGAGGTTCTAGTTTTC	1	4.9662	53	27.0833	1.95692	35.5828	-1.60265
TTCAAAGCTTTGATGGTGT	2	4.99642	68	40.3467	1.68539	35.4958	-1.83115
CAAATCCGCTAACCAAGCT	1	4.98864	60	33.469	1.7927	35.0234	-1.60716
CTTCCCAAAAGCTAAAGTT	1	3.67201	26	6.78082	3.83434	34.944	-1.30074
ACTCAATGATACACAAAGA	1	3.3705	24	5.71429	4.2	34.442	-1.21506
GTGTTACTACATGCAATGT	1	4.9958	66	39.3904	1.67553	34.0647	-1.6086
TATGCAATGCAATCGTATA	1	4.80541	40	17.1429	2.33333	33.8919	-1.56974
CTAGAGTCATACATTTTTG	1	2.85293	21	4.28814	4.89723	33.3621	-1.04835
GTACTACAACGACATGATA	5	4.91661	45	21.9027	2.05455	32.4025	0.0840931
CTCAAAGCTTTGATGGTGA	3	5	172	142.668	1.20559	32.1592	-1.53248
TATACGATTGCATTGCATA	2	3.42744	23	5.89916	3.89886	31.2957	-1.07733
TTTAAAAGCCTAAGTATTA	1	4.95832	49	25.8732	1.89385	31.2921	-1.60107
TTTTTTGGTTTTGGATCTC	1	3.63531	24	6.6383	3.61538	30.8448	-1.29069

Table A.168: Top 25 Words by $O * ln(\frac{O}{E})$ Score, Length 20

Word	**S**	E_s	**O**	**E**	$\frac{O}{E}$	$O * ln(\frac{O}{E})$	$S * ln(\frac{S}{E_s})$
TTGTTTCCTTGTTAGAAGAA	1	3.02668	35	4.72303	7.41049	70.1014	-1.10747
ACGGATCTTAAAAGCCTAAA	1	4.99129	78	35.0364	2.22626	62.4251	-1.60769
ATCCTTGTTAGAAGATACAT	1	4.98333	73	31.2087	2.33909	62.0327	-1.6061
TGGACTTTGGCTACACCATG	4	4.86308	55	19.1045	2.87891	58.1576	-0.781512
TTTGTTAGAAGATACAAAGA	1	4.98458	66	31.6674	2.08416	48.4682	-1.60635
CCGGATCTTAAAAGCCTAAG	4	4.99997	108	69.1227	1.56244	48.1947	-0.892548
TTCCTTGTTAGAAGATACAA	5	5	259	218.174	1.18713	44.4277	2.22045e-15
CGGACTTTGGCTACACCATC	4	5	123	87.8806	1.39963	41.3533	-0.892573
TAACCCTAAACCCTAAACTT	1	2.01545	18	2.60274	6.91579	34.8085	-0.700844
TAAACCTAAACCCAAAACCA	1	4.9871	59	32.7182	1.80328	34.7868	-1.60686
TTAAACCCTAAACCCTAACT	1	4.88325	44	20	2.2	34.6921	-1.58581
ATCCGCACTCAACCAGACAA	5	4.60041	35	13.1881	2.6539	34.1611	0.416458
TTTCCCAAAAGCTAAAGTAG	1	4.55497	34	12.6034	2.69767	33.7413	-1.51622
TAAATCCTACTTTAGCTTCC	2	4.04149	28	8.49558	3.29583	33.3945	-1.40693
CATTTTCCCGCCAAAAACGA	4	2.63553	20	3.79259	5.27344	33.2536	1.66884
AAATTTAGATATCAAATCTT	5	4.91891	45	22.0612	2.03978	32.0778	0.081759
CTTGTTAGAAGATACAAAGC	5	5	205	176.989	1.15826	30.1192	1.17351e-12
TAATTTAGATATCAAATCTA	5	4.96281	49	26.5306	1.84692	30.0625	0.0373307
AAAGGTCAAAATTTGGAGTC	2	4.67504	34	14.3158	2.375	29.4099	-1.69818
CTCATATTTCACTCTGAAAT	1	2.26198	17	3.04211	5.58824	29.2513	-0.816243
ATTCCCAAAAGCTAAAGTAA	1	4.99598	63	39.6466	1.58904	29.1772	-1.60863
AAAACCTTAAACCCTAAACA	1	4.9901	57	34.2826	1.66265	28.9795	-1.60746
AAGATTTGATATCTAAATTT	5	4.87815	40	19.759	2.02439	28.2107	0.123364
TAAGGTCAAAATTTGGAGTA	1	4.98771	55	33	1.66667	28.0954	-1.60698
CTCTCTCTCTCTCTCTCTCT	5	5	2218	2190.45	1.01258	27.7194	0