

White Paper

Xtreme Compression: Beyond the State of the Art

August 30, 2010

EXECUTIVE SUMMARY

Topic. Xtreme Compression is a set of four lossless methods that compress large structured databases for fast real-time query processing or archiving. This technology breaks new ground by systematically exploiting multidimensional table structure, taking full advantage of data semantics, accommodating heterogeneous data types, and predicting bidirectionally. That makes these methods far less subject than conventional ones to compressibility limits imposed by information theory, so they achieve unequalled compression.

Motivation & opportunity. Database volumes are growing faster than Moore's law, but the state of the art of database compression is not keeping pace. In the absence of systematic methods designed for databases, decades-old, one-dimensional methods intended for long-obsolete hardware are instead having to be brought to bear *ad-hoc* on database data. That role mismatch costs performance, but it creates opportunity for superior methods such as these.

Benefits. Database sizes and access times reduce by factors beyond the reach of the conventional methods now widely used. Compression ratios are achievable beyond thresholds at which otherwise-infeasible operational goals, products, and services then become attainable.

Applications. Xtreme Compression targets applications having relatively static structured data. It is ideal for large multidimensional databases and real-time, read-intensive query workloads, such as in memory-centric DBMSs, data mining, data warehousing, business intelligence, statistical databases, analytics, and the like. It is also ideal for devices constrained by memory, power dissipation, or bandwidth.

Use & implementation. Database data structures are individually compressed and then used in compressed form. Decompression is encapsulated and invisible above the lowest software levels, thus setting a distinct level above which the host system remains unaffected and unchanged, and preserving existing access paths and all higher-level design aspects.

Technology application. Xtreme Compression is custom engineered to client-specific requirements and then, typically, integrated into the lowest levels of the access and retrieval software mechanisms of the client's database management system.

Practicality. This technology's systematization brings to a minimum the number of discrete methods having to be implemented, optimized, and maintained to compress databases, making it cost-effective and practical.

1 PROBLEM & SOLUTION

Problem: Unmet Demand and Opportunity

A demand has arisen that technology has failed to answer. With the proliferation of specialty database management systems, new technical strategies, and very large multidimensional databases has come a new urgency, and opportunity, for fast and lossless data compression methods that perform better on database data than what is now being used.

That demand persists not only because database volumes have been growing faster than Moore's law, but more to the point, because the state of the art of database compression has not been keeping pace. Nor has the practice of database compression begun to keep pace with the technology advances that provided those systems, strategies, and databases. One reason is a lack of instructive principles to supply insight and guidance; whatever the others, systematic and innately multidimensional methods designed expressly for databases are rare.

In their absence, decades-old, 'industry standard', conventional data compression methods, developed for one-dimensional data and long-obsolete hardware, are instead having to be brought to bear *ad-hoc* on the problem of database compression, typically by applying one method to each table field. That orthodox but all-too-common approach has two serious shortcomings:

- ! *Information loss.* The methods have no means to communicate one to another information about data redundancy, so much of that is lost
- ! *Role mismatch.* One-dimensional methods are morphologically mismatched to their roles because they cannot model table data at the appropriate levels of abstraction, so they fail to exploit the data's dimensional structure

Those shortcomings make the orthodox approach less effective than an enterprise-specific one would be, but in doing so they create opportunity. They also suggest design goals for new, systematic, and innately multidimensional database compression methods that would break new ground and get beyond the state of the art.

Solution: Invent New Methods

Xtreme Compression does just that. It is an innovative, powerful, and proven set of four algorithmic and structural methods developed to compress large multidimensional databases for fast real-time access in data-intensive applications. It leverages the benefits of the multi-million-dollar government research & development program that developed the technology from which it evolved.

Xtreme Compression employs an instructive and goal-aligning principle. The methods use sophisticated data modeling that makes them far less subject than conventional methods to compressibility limits imposed by information theory. Together, they simultaneously achieve absolute transparency, fast decompression, and compression ratios well beyond the reach of the decades-old conventional methods now widely used. Most significant, these methods can often achieve compression ratios beyond thresholds at which otherwise-infeasible operational goals, products, and services then become attainable.

Each method is specifically engineered for one class of data and data structures:

- ! *Attribute vector coding*, the technology's centerpiece, compresses all fields of multidimensional database tables except some containing text strings.
- ! *Wordencoding* compresses the remaining text strings.
- ! *Repopulation* compresses hash and collision tables.
- ! *Superpopulation* compresses index tables, lists, arrays, zerotrees, and the like.

Data structures are individually compressed and then used in compressed form. Decompression is encapsulated and invisible above the lowest software levels, thus setting a distinct level above which the host system remains unaffected and unchanged.

This technology targets applications having relatively static structured data. It is ideal for large multidimensional databases and real-time, read-intensive query workloads, such as in memory-centric DBMSs, data mining, data warehousing, business intelligence, statistical databases, analytics, and the like. It is also ideal for devices constrained by memory, power dissipation, or bandwidth.

Along with the traditional advantages of reduced database size, Xtreme Compression provides faster access by doing the following:

- ! Concentrating more information into every physical word read from memory and processed by the CPU to reduce the influence of the 'von Neumann bottleneck' and effectively increase the capacity of the in-memory database
- ! Storing more real information in internal caches during query to improve data locality and cache hit rates, especially with newer processor technologies
- ! Increasing effective disk cache size to retrieve more real information from hard disk per read operation
- ! Allowing storage on faster but more costly media (i.e., RAM vs hard disk, hard disk vs CD or DVD, local hard disk vs network, etc.)
- ! Permitting more access data structures and paths for a given amount of storage
- ! Reducing transmission time over bandwidth-restricted links, networks, and other infrastructure components that are expensive to scale
- ! In some applications, reducing collision frequency by setting the hash table load factor at a lower value than what would otherwise be practical

Another benefit in some applications is reduced device power consumption.

Xtreme Compression is custom engineered to client-specific requirements and then, typically, integrated into the lowest levels of the access and retrieval software mechanisms of the client's database management system.

2 BACKGROUND

Principles vs Practice

More than half a century has passed since the advent of information theory as a formal intellectual discipline and data compression as a field of practice. Claude Shannon's seminal paper *A Mathematical Theory of Communication* marked the birth of information theory in 1948. Four years later, David Huffman's treatise on entropy code construction, which referenced work from 1949, sparked the development of practical encoding methods that continues to this day.

That came about before the arrival of computer databases. Back then, the most common usage for large data volumes was communication. It typically involved the first-in, first-out transmission and reception of structurally simple data. The storage and random retrieval of discrete multidimensional data using computers and memory would not come about until later.

As one would expect, a number of factors shaped the theory's scope and focus and, more to the point, the range of its application to data compression and other fields of practice. Among them were no doubt the interests of funding agencies, the opinions of influential researchers, the launch of *Sputnik*, the hardware costs of the times, the operational metaphors of the era, and the nature of the epoch's data. Sixty years later, the published literature illustrates that database compression has rarely ever been within the boundaries.

For that and similar reasons, information theory, in particular the noiseless coding theorem, has less relevance and value than what one might expect toward the modern problem and practice of designing compressed databases. The theorem is not instructive in that way, providing no useful principles and little in the way of insight or guidance. The following discussion explains.

The Need for Principles

Symbol definition. Three subtle implications from information theory bear on a fundamental design matter. That is how to define the symbols when the design objective is to maximize compression. This discussion explains how information theory motivates pursuing a particular set of design subobjectives, but the theory fails to provide any insight or guidance on what the designer should do to meet them.

- ! *Exclusive scope.* Information theory concerns sequences of symbols, not the nature of symbols themselves. By doing that, it leaves unanswered the pivotal questions of what a symbol can and should be. By imposing no constraints on symbol form or content, information theory puts the matter squarely in the hands of the designer.

In real-world database tables, the amount of redundancy a symbol holds will depend on the amount of uncompressed data it represents, increasing monotonically with that. At the same time, the various occurrences of redundancy within individual table records will not in general be statistically independent. Those two considerations, taken together, suggest that the designer choose as symbols the largest divisions of original data that can practically be dealt with in software.

- ! *Organized redundancy.* There is also a spatial frequency implication. The organized redundancy that is characteristic of table data will in general have strong presence at the spatial frequencies that correspond to individual table records. For that

reason, the designer should expect that defining symbols at the record level will yield particularly good results, as will defining them as integer multiples of records.

- ! *Entropy bound in the limit.* The third implication arises out of the well-known equation for computing the entropy of the source, which in turn bounds compression efficiency. In the asymptotic case where the table data are represented by a single symbol, the computed entropy is zero regardless of the nature of the symbol. Although of course that result is only an asymptotic estimate, it is nonetheless the only quantitative information that information theory provides that bears on the bound on compression performance in the limit where a very few very large and complex symbols represent the data.

So information theory motivates defining highly complex symbols that represent records or multiples of records and large amounts of original data, and using few of them. But that design direction and those subobjectives are the only guidance that information theory contributes to the matter. The ultimate design problem of how to get that done is left to the designer. Information theory is not constructive here, shedding no further light.

Symbol correspondence and prediction directionality. Shannon's work centered on a transmission model that used a replacement scheme. It preserved one-to-one correspondence between uncompressed and compressed symbols. Contrast that to a database table persisting in memory and available in its entirety to a decompression mechanism: all that is required in principle is that any part of the compressed data be transformed back into the original data it represents. How information is distributed among compressed symbols does not matter.

What does matter is that, once the compressed database design is unencumbered by any need for symbol correspondence, two new avenues open for better compression:

- ! A class of compression techniques become available to the designer that excel at sharing information among widely distributed symbols.
- ! Compressed symbols can be arranged for *bidirectional* prediction – i.e., predicting the present from the future and the past, rather than from the past alone. Bidirectional predictions can be more accurate than one-directional ones because twice as much information is available on which to base them, and because, being symmetric, they do not suffer from prediction lag.

Information theory does not address this issue.

Information vs meaning. Rather than to coin a new word, or to select one carrying different connotation, Shannon chose to reuse the word 'information' in a special sense for a special purpose: analyzing communication. He computed a probabilistic sum called the entropy of the source, equated it to information, and set that as the limit to which encoding could compress data. Thus he gave information a second definition, this one purely quantitative and statistical.

Shannon, cognizant of the divergence of this new parlance from the term's historic usage, pointed out, "semantic aspects of communication are irrelevant to the engineering problem." Warren Weaver, also mindful of the danger of confusion, more succinctly but just as ominously forewarned in his explication, "information must not be confused with meaning."

Coming early as it does in Shannon's second paragraph, that distinction is foreboding. Equally disquieting is the portrayal of random sequences as data high in information content, whereas in actuality, they are just noise. Together, those caveats portend a much more central role for semantics in the practice of database compression than for Shannon's vacuous 'information'. They also call into question how valuable the theory can ever be to the practice. On the other hand, if one is astute, they do hint at where, and where not, to concentrate engineering effort.

The discord arises from disparity of purpose. Data have structural and statistical properties that, while conveying meaning, are useful for the purpose of compressing data, but useless for Shannon's purpose of analyzing communication.

Sixty years later, one can appreciate that databases are intrinsically semantic, and realize that, since they can be treated as having meaning, they should be. Taking full advantage of meaning allows one to model the data at the highest levels of abstraction, and so is invaluable. But how to do that falls outside information theory's purview, so one must look elsewhere for guidance.

Scope of compressibility limit. By categorically excluding semantics, information theory avoids addressing what for compression is the all-important issue: abstraction, the function of data modeling. More to the point, information theory does not govern how data can be modeled before they are encoded. Clearly then, in database compression, where one or more modeling steps precede the encoding step, *the limit on compressibility applies only to the encoding, not to the modeling.* Figure 1-1 illustrates that crucial point.

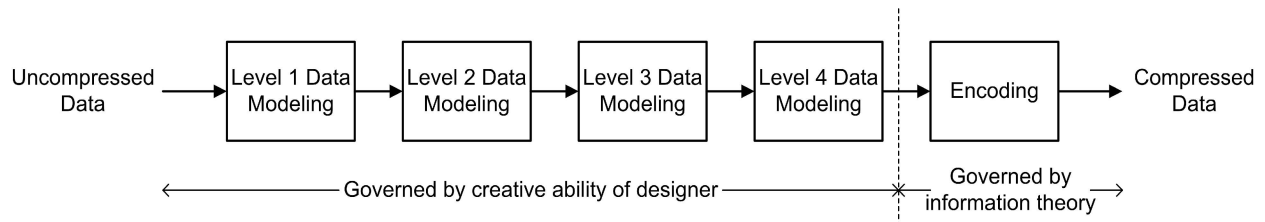


Figure 1-1. Differentiation of domains of designer ability and information theory.

That differentiation becomes all the more profound in light of two other facts. First, for table data, modeling is naturally more complex than encoding due to dimensional structure. Second, and more fundamental, *no principle or theory governs the succinct codification of meaning.*

Consequently, for sufficiently large and complex tables, the model can take virtually any form the designer can imagine. Its capability is limited only by the designer's creative ability. That means table data should ultimately be compressible almost without limit as long as the designer can invent good enough models. Here again the theory contributes little insight, so that inventing becomes the challenge, the goal, and the promise of Xtreme Compression.

The Evolution of Lossless Data Compression

Much has changed over the last half-century. Computer databases have overtaken communication as the primary usage of large data volumes. More recently, as mentioned earlier, specialty database management systems and new technical strategies, each having its own requirements for fast access to structured discrete data, have proliferated, as have the large databases themselves.

The available literature documents how the practice of lossless data compression has evolved over that period. It is noticeably deficient in topics more pertinent to database compression. Specifically, the literature reveals the following:

- ! *Data classes and systematization.* Three classes of target data – unstructured text, still pictures, and indexes – dominate the literature. Nonsystematic approaches, such as using a plurality of existing techniques and *ad-hoc*, data-specific ones, are found, but there is little that treats databases as databases to better exploit the nature of their data. Innately multidimensional and systematic approaches are rare.
- ! *Regime.* Most published techniques assume the data, once compressed, will either be archived for decompression at a later time, or else transmitted first-in-first-out over a channel and then decompressed symbolwise. Fewer focus on real-time access to memory-resident compressed data.
- ! *Features and compromises.* Methods from the literature very often feature built-in facilities for adapting to changing data statistics or for updating data while compressed. Such features are usually of lower priority in database applications. Providing them generally sacrifices performance, so whether and how to implement them should be considered in the context of individual client requirements.
- ! *Originality.* Small improvements far outnumber fundamentally new ideas. There is considerable emphasis on variations of the practical, popular, and adaptable LZ class of methods, now about three decades old, and on implementations of arithmetic coding, ever-efficient and now over two decades old.
- ! *Generality of purpose.* Most published methods strive to be useful for all data of a given type, and so are optimized and evaluated that way. One reason is the obvious market advantage of broad applicability. Another is the understandable lack of reader interest in methods for data that appear too esoteric. Unfortunately, general-purpose methods inevitably compromise performance.

By contrast, requirements of individual clients, applications, and enterprises are always specific, not only in factors like data structure, semantics, and statistics, but also in the need for prediction mechanisms that are morphologically suited to the dimensional structure of their data.

From the above discussion it is clear that the practice of database compression should be much more art than science.

More to the point, over the last half-century, the research and development community appears to have contributed little needed science – principles, insight, and guidance – to database compression. From the opposite view, some technology needs specific to database-related industries and fields are not being well met by academia or by industry. So the need for database compression remains beyond the state of the art. As long it does, there will be opportunity for advances like Xtreme Compression.

On Requirements and Expectations

These discussions have shown that the practice of database compression has an unmet need for an instructive principle to provide insight and guidance. They suggest that, due to the lack of applicable methods in the literature, much algorithmic new ground will have to be broken.

Finally, these discussions lead one to expect that newer methods that make use of complex symbols, share information efficiently among distributed bits, predict bidirectionally, take full advantage of semantics, incorporate sophisticated data modeling, systematically exploit dimensional structure, focus on fast real-time access, appropriately set priorities for adaptivity and update capability, introduce fundamentally new ideas, and address client-specific requirements would outperform the decades-old, one-directional, one-dimensional, 'industry standard' conventional methods now widely used, and so get beyond the state of the art.

Not surprisingly, they do.

3 THE APPROACH TO COMPRESSING DATABASES

The Exchange Principle

Xtreme Compression introduces an instructive principle called the *exchange principle*. It has enormous practical value to the practice of compressing databases because it guides the database designer to work in a particular way that experience has proven gets the desired results. The exchange principle is motivated by points made previously, and by the following observations drawn from experience:

- ! The value of any stored information lies in its retrieval, not in its storage.
- ! Information is stored for the future but retrieved for the present; for that reason alone, retrieval is usually much more time-critical than storage.
- ! The time required to retrieve information from memory depends on the amount of data holding that information, not on the amount of information held in the data.
- ! The trend toward more and larger data caches has consequences particular to data compression because it affects the historical need to compromise size with speed.
- ! Files, tables, and data sets do not have entropy; only models of data have entropy. For data with sufficient algorithmic complexity, interrelationships, functional dependencies, and statistical correlations, the capabilities of the data model are limited only by the creative ability of the designer. No principle or theory restricts the capability of such models.
- ! During database design, function and algorithmic complexity can be exchanged between the realms of compression and decompression to a degree limited only by the creative ability of the designer. No principle or theory restricts that exchange.
- ! During database design, function and algorithmic complexity can be exchanged between data modeling and encoding to a degree limited only by the creative ability of the designer. No principle or theory restricts that exchange.

Recognition of those truths leads to the exchange principle:

During design, moving function and algorithmic complexity from decompression to compression and from encoding to modeling can simultaneously increase compression effectiveness and decrease decompression time.

The exchange principle *aligns goals*. It eliminates having to compromise size with speed; for the first time, those otherwise-competing performance goals can go hand in hand. It explains why these methods are so asymmetric, i.e., more complex in compression than in decompression. Also, the exchange principle motivates a modeling-to-encoding complexity imbalance, the specific factor that makes these methods far less subject than conventional ones to compressibility limits imposed by information theory.

Although the exchange principle contributes to the design process much-needed insight and guidance, it is not in any way constructive. Neither is information theory. That leaves the designer's creative ability as the lone source of intellectual spark and fuel for devising the methods that form the practice of database compression. That is one reason the state of the art lags by decades the technologies that led to the unmet demand for better performance. It also explains why the practice is much more art than science.

Use and Implementation

Xtreme Compression provides four independent, transparent, and application-specific methods, and applies one or two to each database data structure. Each structure is then stored and accessed in compressed form. Individual table records, blocks of records, or index table locations, as required, are decompressed on the fly during query execution. Where required, some query operations can be performed on compressed or partially-decompressed data.

The methods' absolute transparency, separately and together, preserves the existing access paths and all higher-level aspects of the database design, thereby setting a distinct level above which the host database management system's software remains unaffected and unchanged.

4 THE METHODS

Xtreme Compression encompasses the four methods described briefly below. In practice, additional more application- and client-specific techniques are employed that are not discussed.

Method 1: Attribute Vector Coding

Attribute vector coding is a vector transform method for compressing multidimensional database tables. It is tuple-oriented and semantics-aware. It breaks new ground by capturing and exploiting the innumerable relationships, functional dependencies, and statistical correlations in the data *without having to solve the intractable problem of explicitly identifying and defining them*. It achieves unequalled compression because it systematically models data at the highest levels of abstraction, across dimensions and data types. That makes this method far less subject than conventional methods to compressibility limits imposed by information theory.

Attribute vector coding recovers an empirical understanding of the processes that created the data through statistical analysis, and then strives to emulate those processes through functions

embodied in software and codified in data. To that end, it captures and exploits prior knowledge *regardless of whether it can be explicitly identified and defined*. That prior knowledge has two parts: intra-tuple correlations, from other fields within the record, and inter-tuple correlations, from fields in other records. From those, structured predictions of field values are computed and expressed, together with differences between them and actual values, through a system of functions and coefficients. Those functions, and the scheme for encoding them into symbols, decorrelate across dimensions and data types simultaneously.

Of course, the advantages of high-level abstraction would all be for nought if attribute vector coding were not cost-effective to use. That is why one other aspect is vital: flexibility. It is what allows attribute vector coding, when used together with wordencoding, to systematically accommodate all table fields regardless of data type, cardinality, skew, sparsity, field order, or field width. That systematization brings to a minimum the number of discrete methods having to be implemented, optimized, and maintained to compress table data, thereby helping to make attribute vector coding, above all, practical.

Method 2: Wordencoding

Wordencoding is a 0-order (context-independent) variable-to-variable-length algorithm for compressing text strings, hereinafter called words, in database table record fields. It achieves compression close to the 0-order source entropy without sacrificing speed. It does that by providing an efficient way to maximize effective combined data locality over three areas: the compressed record fields, the lexicons holding the words, and the lexicons' access data structures.

Wordencoding recognizes that word frequencies will represent great statistical disparity, and it *accommodates statistical disparity with algorithmic diversity* – the systematic use of multiple techniques. Further, by decomposing words into fragments and compressing each fragment separately, it recognizes that uncommon words are less compressible than common ones, and that they often consist of two more common and therefore more compressible fragments. Doing that deals explicitly with the structure and statistics of the data by recognizing that redundancy in text strings exists at different levels of granularity.

Wordencoding uses two lexicons constructed by sectioning by word length and storing all lexicon words of each length together. The motivation is the observation that the ratio of discrete words to discrete word lengths will always be high. The benefits are twofold: reducing the lengths of the lexicon indexes stored in database table record fields after compression, and allowing access through data structures compact enough to be kept in cache memory. Thus, exploiting word length statistics is significant to the compression and the decompression.

Method 3: Repopulation

Repopulation is a structural method for compressing the common configuration of access data structures consisting of separate hash and collision tables, and handling collisions – sequences of database record numbers – through external chaining. Unlike most compression methods, repopulation is not a replacement scheme; it draws on no information-theoretic concepts.

Repopulation is mechanistic; it operates like a chess-playing automaton. It populates hash table locations that would otherwise be empty with parts of collision strings – sets of three or more locations sharing a hash address – that would otherwise occupy memory. It is transparent, imposes no new requirements on the hash functions, preserves fast real-time random access,

and does not degrade collision statistics. Repopulation simultaneously achieves the access speed advantage of a low hash table load factor and the table compactness of a high one, thus avoiding the historical compromise of speed with size.

The presence of empty hash table locations while record numbers in collision strings need to be stored suggests a particular avenue for compression. That is to move, or *repopulate*, individual collision strings out of the collision table and into otherwise-empty hash table locations.

The access data structures are compressed as a unit after being generated from database table record terms, as a final processing step before storage in memory. During access, collision strings are retrieved and reassembled in real time on a term-by-term, string-by-string basis within the queries, just prior to reading the table records.

Method 4: Superpopulation

Superpopulation is a variable-to-variable-length algorithm that compresses index tables, lists, arrays, zerotrees, and similar types of data. It systematically accommodates and instantaneously adapts to wide local variations in data statistics. Depending on the application, superpopulation may be used either by itself or in conjunction with repopulation. That is because the two methods operate on different aspects of data representation.

The motivation for superpopulation is the observation that distributions of values in access data structures are often far from random. They can be highly correlated due to the nature of the processes that generated the data, and they usually have areas of high and low correlation. In a typical index table application, superpopulation decomposes each collision string into a sequence of adjacent substrings, each classified either as one of two distinct target types or as an incompressible substring. Each type of target is then compressed using one of two target type-specific encoding methods, and is identified in the compressed data by a corresponding escape code.

Superpopulation is implemented as a discrete processing phase during the generation of the data structures. It operates sequentially on all collision strings more than two locations in length.

5 SUMMARY

There are three important requirements for methods that compress large structured databases for fast real-time access. They are transparency, decompression speed, and compression effectiveness. The transparency requirement just prevents using a class of conventional methods, i.e., lossy ones. Access speed is designed into the methods primarily through the guidance of the exchange principle. Unmatched compression effectiveness is achieved through sophisticated and complex data modeling; it is the distinguishing advantage of Xtreme Compression.

This document has discussed the following reasons why an enterprise-specific design approach that focuses on client requirements, and uses newer technology engineered according to an instructive principle, would naturally be expected to meet those requirements:

- ! From the beginning, database data are treated as database data with regard to dimensional structure, data semantics, heterogeneity of data types, and prediction directionality; that drives all of the engineering design and analysis.

- ! The database design is unencumbered by any need for symbol correspondence; that makes available a class of techniques that could not otherwise be used, and it allows making bidirectional predictions, which can be more accurate.
- ! Recognizing semantic content allows understanding and modeling data at higher levels of abstraction than would otherwise be possible, an overwhelming advantage.
- ! General-purpose methods invariably compromise performance because of how they are optimized and evaluated.
- ! The exchange principle aligns the otherwise-competing performance goals of compression ratio and decompression speed, eliminating historical compromise.
- ! Attribute vector coding, the centerpiece of the technology, captures and exploits the innumerable relationships, functional dependencies, and statistical correlations in the data without explicitly identifying and defining them.
- ! By design, two methods, wordencoding and superpopulation, accommodate statistical disparity in the data through algorithmic diversity.
- ! Compressibility limits imposed by information theory apply only to encoding, not to data modeling. Furthermore, with respect to the modeling, no principle or theory governs the succinct codification of meaning. Consequently, for sufficiently large and algorithmically complex database tables, the capabilities of the model are limited only by the creative ability of the designer. No principle or theory governs the design of such models.

After considering the above, it is natural to expect Xtreme Compression to outperform the decades-old compression methods now widely used, and so get beyond the state of the art.

Not surprisingly, it does.