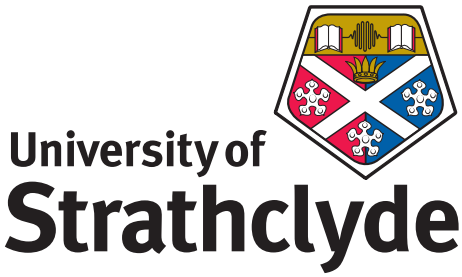


SIMILARITY SEARCH USING INFORMATION THEORETIC MEASURES

ROBERT GEORGE MOSS

A thesis presented to the department of Computer and Information Sciences in fulfilment of the requirements for the degree of Doctor of Philosophy

January 7, 2016

DECLARATION

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Glasgow, January 7, 2016

Robert George Moss

For my daughter, Alice

ABSTRACT

ACKNOWLEDGMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

1

INTRODUCTION

Many modern data collections defy the traditional storage mechanisms of primary key based relational databases. With no natural ordering, it becomes either impossible or undesirable to index collections of this type. For example, with a corpus of semi-structured data, or images on the world-wide-web, exact-match solutions, which have typified traditional databases, provide inadequate and often meaningless solutions.

As the number of such collections grow, there becomes a pressing need to address the problem of search. In many cases, similarity searching provides the answer.

1.1 THESIS OUTLINE

Having introduced the notion of similarity search in general terms, we now outline what follows in this thesis. Part 1 introduces the mathematical preliminaries that underpin the work in subsequent parts. It consists of a single chapter which covers aspects of distance, probability, and information theory that are pertinent. This is followed in Part 2 with three chapters that deal with combining distance, probability and information theory. First, by introducing various methods of computing distance with information in chapter 3; Secondly, by narrowing the focus specifically to Structural Entropic Distance in chapter 4 and generalising it to the notion of multi-way distance; and thirdly, by exploring efficient means of evaluating Structural Entropic Distance. After this, follows the final part, Part 3, which investigates the use of Structural Entropic Distance. It consists three further chapters: Chapter 6, investigates the performance of Structural Entropic Distance as a distance metric used with Similarity Search indexing techniques; chapter 7, explores a modification to Structural Entropic Distance to perform the traditional Information Retrieval task of searching for documents relevant to a set of keywords. Finally, in chapter 8 we make use of the

multi-way distance generalisation to perform classic data-mining techniques such as clustering, classification, and outlier detection.

Part I

MATHEMATICAL PRELIMINARIES

The concept of similarity is a fairly intuitive one. Objects that are similar have features in common, while dissimilar objects have less. Understanding this from a mathematical point of view requires that this be somehow measured and assigned a number. Fortunately, another concept fits nicely with similarity – distance. We all already use it colloquially, e.g. "A close match". In this part, we show how distance as the sole measure of similarity has all of the properties required to make an effective search paradigm. We describe the mathematical abstraction of metric spaces; how spaces can be divided to produce efficient storage structures; and common queries that make similarity searching worthwhile.

2

DISTANCE, PROBABILITY SPACES AND INFORMATION THEORY

In this chapter we review the basics of information theory starting with a brief recap of probability, then introduce information, and finally derive entropy as defined by Shannon in his seminal work[**?**] to serve as a foundation for later chapters.

2.1 DISTANCE

2.1.1 *Metric spaces*

A metric space is simply a set of objects which have distances between them. Our own understanding of the physical world informs our intuition for the postulates of a metric – there cannot be a negative distance; the distance between points is the same in both directions; if there is no distance between two objects they must be be at the same point (be the same object) – likewise, there is no distance between two identical objects; and finally, there is no shorter distance between two objects by taking a detour through a third object.

These axioms are defined formally as follows:

Definition 1. *A metric space* M *is an ordered pair* (X, d) *where* X *is a set and* $d : X \times X \to \mathbb{R}$ *is a metric that defines the distance between elements in* X*. Furthermore,* d *must satisfy the following properties:*

$\forall x, y \in X,$	$d(x, y) \geqslant 0$	(positiveness)
$\forall x, y \in X,$	$d(x, y) = d(y, x)$	(symmetry)
$\forall x, y \in X,$	$d(x, y) = 0 \Leftrightarrow x = y$	(reflexitivity)
$\forall x, y, z \in X,$	$d(x, z) \leqslant d(x, y) + d(y, z)$	(triangle inequality)

2.1.2 *Vector spaces*

Vector spaces are more specific than metric spaces, since they allow the elements of the space to be added or scaled to produce new elements.

Definition 2. *A metric space* $M = (V, d)$ *is also a vector space if* V *is closed under addition and multiplication operations. Furthermore, the following axioms must hold for addition:*

$\forall \mathbf{x}, \mathbf{y} \in V,$	$\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$	(commutativity)
$\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in V,$	$(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$	(associativity)
$\forall \mathbf{x} \in V,$	$\mathbf{0} + \mathbf{x} = \mathbf{x}$	(identity)
$\forall \mathbf{x} \in V,$	$\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$	(existence of additive inverse)

The following axioms must hold for multiplication:

$\forall \mathbf{x} \in V, \forall a, b \in \mathbb{R},$	$a(b\mathbf{x}) = (ab)\mathbf{x}$	(associativity)
$\forall \mathbf{x} \in V,$	$1\mathbf{x} = \mathbf{x}$	(identity)

And the following laws of distributivity also:

$$\forall \mathbf{x} \in V, \forall a, b \in \mathbb{R}, \qquad (a+b)\mathbf{x} = a\mathbf{x} + b\mathbf{x} \qquad \text{(scalar sums)}$$

$$\forall \mathbf{x}, \mathbf{y} \in V, \forall a \in \mathbb{R}, \qquad a(\mathbf{x}+\mathbf{y}) = a\mathbf{x} + a\mathbf{y} \qquad \text{(vector sums)}$$

As a further refinement we now define a Normed Vector Space which adds the concept of magnitude to the elements of the space.

Definition 3. *A normed vector space* $N = (V, \|\cdot\|)$ *where* V *is a vector space and* $\|\cdot\|: V \to \mathbb{R}$ *a normalisation function. Furthermore, the normalisation function must satisfy the following three properties:*

$$\forall \mathbf{x} \in V, \qquad \|\mathbf{x}\| > 0 \Leftrightarrow \mathbf{x} \neq \mathbf{0} \qquad \text{(positivity)}$$

$$\forall \mathbf{x} \in V, \forall a \in \mathbb{R}, \qquad \|a\mathbf{x}\| = |a|\|\mathbf{x}\| \qquad \text{(positive scalability)}$$

$$\forall \mathbf{x}, \mathbf{y} \in V, \qquad \|\mathbf{x}+\mathbf{y}\| \leqslant \|\mathbf{x}\| + \|\mathbf{y}\| \qquad \text{(triangle inequality)}$$

As a final refinement to vector spaces, let us now add the concept of inner product.

Definition 4. *A inner product space* $I = (N, \langle\cdot,\cdot\rangle)$ *where* $N = (V, \|\cdot\|)$ *is a normed vector space and* $\langle\cdot,\cdot\rangle : V \times V \to \mathbb{R}$ *an inner product function. Furthermore, the inner product must satisfy the following three properties:*

$$\forall \mathbf{x}, \mathbf{y} \in V, \qquad \langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle \qquad \text{(symmetry)}$$

$$\forall \mathbf{x}, \mathbf{y} \in V, \forall a \in \mathbb{R}, \qquad \langle a\mathbf{x}, \mathbf{y} \rangle = a\langle \mathbf{x}, \mathbf{y} \rangle \qquad \text{(scalar linearity)}$$

$$\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in V, \qquad \langle \mathbf{x}+\mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle \qquad \text{(vector linearity)}$$

$$\forall \mathbf{x} \in V, \qquad \langle \mathbf{x}, \mathbf{x} \rangle \geqslant 0 \qquad \text{(positive definiteness)}$$

2.1.3 *Distance Metrics*

Given these mathematical abstractions, we might ask what kind of distance functions exist that satisfy these axioms and what are they good for. Let us start with a simple example, we would like to measure the similarity of two strings of characters. This could be because we are building a spelling checker that might give suggestions of the most likely intended word. It might also be required for a biologist working on DNA sequences that needs to find commonalities to identify some gene's function.

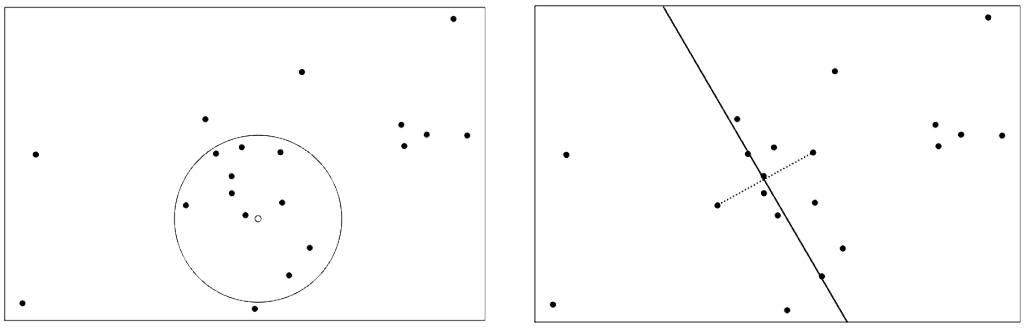
2.1.4 *Space partitioning*

As is typical with other types of storage system, partitioning the space allows for more efficient searching by excluding whole subsets from the search space, thus reducing effort required. In contrast to other Spaces, (e.g. Vector Spaces), Metric Space and by implication the associated distance function contains very little information about the actual layout of the space. It is, however, possible to use the properties of the metric, and in particular, triangle inequality to divide up the space in a principled way.

[**?**] introduced two techniques, *Ball decomposition*, and *Generalised Hyperplane decomposition* which have formed the basis of most index design to date; while *Excluded Middle*, which extends *Ball decomposition*, has been suggested by [**?**].

2.1.4.1 *Ball decomposition*

Definition 5. *Let* d_m *be the median of* $\{d(p, o_i), \forall o_i \in X\}$ *for some pivot* $p \in S \subseteq X$. *Partition* S *into two subsets* S_1 *and* S_2 *such that* $S_1 = \{o_j : d(p, o_j) \leqslant d_m\}$ *and* $S_2 = \{o_j : d(p, o_j) > d_m\}$



(a) Ball Decomposition. (b) Generalised hyperplane Decomposition.

Figure 1: Ball Decomposition – the area inside the ball is one partition, the area outside is another. Generalised hyperplane Decomposition – those points nearest the first pivot form one partition, those nearest the other form another.

2.1.4.2 *Generalised hyperplane decomposition*

Definition 6. *Let* p_1 *and* p_2 *be two arbitary pivots selected from* $S \subseteq X$. *Partition* S *into two subsets* S_1 *and* S_2 *such that* $S_1 = \{o_j : d(p_1, o_j) \leqslant d(p_2, o_j)\}$ *and* $S_2 = \{o_j : d(p_1, o_j) > d(p_2, o_j)\}$

2.1.5 *Dimensionality*

Bellman coined the term *curse of dimensionality* referring to the exponential growth in the number of samples required to estimate a function with a given level of accuracy to its number of variables (or dimensions)[**?**]. This has a direct impact on similarity searching in that the number of points that must be visited to build the result of a query grows exponentially to the number of dimensions in the space. In some cases, data may even be clustered together so tightly that the distance between the nearest neighbour and the farthest neighbour are the same, rendering any result meaningless. In fact, it has been shown that these values converge as the number of dimensions increase

2.1.5.1 *Measuring dimensionality*

Given the difficulty of indexing spaces with high dimensionality, a prudent approach would be to measure this. There is a great volume of work on the analysis of high-

dimensional search spaces. Navarro gives a very elegant analysis of the history of this, along with some cogent insights into directions which could be usefully followed in the future, and bemoans the small volume of current research on the topic.

The Intrinsic Dimensionality (IDIM) of a metric space is a measure defined over a distance histogram produced by the distance metric over the data. It is defined as

$$\rho = \frac{\mu^2}{2\sigma^2}$$

where μ is the expected distance and σ^2 is the variance of the range of the metric over the data collection. An IDIM of n corresponds roughly to the histogram of Euclidean distance over densely-populated n-dimensional Cartesian space. It is usually, but not always, the case that a higher value for IDIM leads to a worse performance of index structures for similarity search.

The authors give a lower bound for the cost of a pivoting operation as

$$\left(\sqrt{\rho} - \frac{1}{\sqrt{2f}}\right)^2 \cdot \ln n$$

where ρ is the IDIM, f is the fraction of the data being returned, and n is the size of the data. However this formula is valid only when $f > \frac{1}{2\rho}$, meaning that for large data sets, and therefore small values of f, it is of limited value. The work uses an underlying assumption that the metric gives a Gaussian distribution of outcomes. More important, we believe, is that no metric distribution obeys a Gaussian distribution close to the origin as distances are by definition lower-bounded at zero, and we later show distributions very close to Gaussian where IDIM comparison is misleading as a predictor of performance.

In the discussion, the authors consider the same region of interest as here, the set of sampled distances which do not allow a median-centred pivot operation to successfully exclude any data from a search.

2.2 PROBABILITY & INFORMATION THEORY

The frequentist view of probability is that there is a set of events that we assume occurs some number of times such that no two events can occur simultaneously.

Definition 7. *let* $E = \{e_1, e_2, \dots e_n\}$ *be a sample space, some* $F \subseteq E$ *be an event space, and* $P : E \to [0, 1]$ *a probability measure where* P *satisfies the following axioms:*

$$P(F) \geqslant 0$$

$$P(E) = 1$$

$$P(\bigsqcup_{F \subseteq E} F) = \sum_{F \subseteq E} P(F)$$

The probability must be positive, since a zero probability means it is certain not to happen you cannot be less certain of an event occurring than that. The second axioms states that the probability that the next event occurring is from the set of all possible events is certain. The third axiom states that the probability of an event occurring from the union of disjoint sets is the sum of probabilities from each of those disjoint sets. Furthermore the following derived properties also hold:

$$P(\emptyset) = 0$$

$$P(F_1 \cup F_2) = P(F_1) + P(F_2) - P(F_1 \cap F_2)$$

$$P(E \setminus F) = 1 - P(F)$$

That is, the probability of an event occurring from the empty set is zero. The probability of an event occurring from the union of two sets is the sum of the probability of it coming from their respective sets minus the probability of the events that have been counted twice. And the probability of an event occurring from outside a set is the probability of the sample space (which from the second axiom is 1) minus the probability of the event space.

A probability distribution assigns a probability to each subset. The proportion of times an event occurs relative to all events that have occurred is the probability of that event.

2.2.1 *Information*

Suppose we have a probability function P, we can measure the information gained by observing an event based on the probability distribution of events.

Definition 8. *Let* $I : [0, 1] \to \mathbb{R}$ *be a monotonic function that measures the information gained by observing an event* e *with probability* $p = P(e)$*, with the following axioms:*

$$\forall p, I(p) \geqslant 0$$

$$I(p) = 0 \Leftrightarrow p = 1$$

$$I(p_1 \cdot p_2) = I(p_1) + I(p_2)$$

The first axiom says that the information function must output a positive value; it makes no sense to have lost information by witnessing an event. The second says that there is no information to be gained by witnessing an event which we know is certain to occur. The third says that the information gained after witnessing two events is the same as gaining the information from witnessing them separately. A commonly used information function which obeys these axioms is $I(p) = -\log_b(p)$ for some base b.

2.2.2 *Entropy*

Given a stream of events whose probability is defined by the some function P the average information gained per event is known as entropy.

Definition 9. *Let* S *be some source emitting a stream of events from the set* $E = \{e_1, e_2, \ldots, e_n\}$ *with associated probabilities* $P = \{p_1, p_2, \ldots, p_n\}$ *respectively.*

There are $N \cdot p_i$ *occurrences of* e_i *after* N *observations, as* N *approaches infinity. So the total information gained is* $I = \sum_{i=1}^{n}(N \cdot p_i) \cdot I(p_i)$*, and the average information per event is therefore:*

$$H(S) = \frac{I}{N}$$

Using $I(p_i) = -\log(p_i)$ *gives,*

$$\begin{aligned} H(S) &= \frac{1}{N} \cdot \sum_{i=1}^{n}(N \cdot p_i) \cdot -\log_b(p_i) \\ &= -\sum_{i=1}^{n} p_i \cdot \log_b(p_i) \end{aligned}$$

Definition 10. *Let* S *and* T *be two sources emitting streams of events from the set* $E = \{e_1, e_2, \ldots, e_n\}$*. Denote the probability of* S *emitting* e_i *and* T *emitting* e_j *as* $p_{i,j}$*. Then the joint entropy of* S *and* T *is:*

$$H(S, T) = -\sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j} \log_b(p_{i,j})$$

2.2.3 *Mutual Information*

Definition 11. *Let* S *and* T *be two sources emitting streams of events from the set* $E = \{e_1, e_2, \ldots, e_n\}$*. Then the mutual information between* S *and* T *is:*

$$I(S, T) = H(S) + H(T) - H(S, T)$$

2.2.4 *Cross Entropy*

Definition 12. *Consider two sources,* S *and* T*, emitting streams of events from the same set* $E = \{e_1, e_2, \ldots, e_n\}$*. Let* S *emit events with probability* $P = \{p_1, p_2, \ldots, p_n\}$ *and* T *emit events with probability* $Q = \{q_1, q_2, \ldots, q_n\}$*. If we model* S *with* T*,* P *is the true distribution and* Q *is the model distribution. After* N *observations, there are* $N \cdot p_i$ *occurrences of* e_i*, as*

N *approaches infinity, so the total information gained is* $I = \sum_{i=1}^{n}(N \cdot p_i) \cdot I(q_i)$*, and the average information gained is:*

$$\begin{aligned} H(S||T) &= \frac{I}{N} \\ &= \frac{1}{N} \cdot \sum_{i=1}^{n}(N \cdot p_i) \cdot -\log_b(q_i) \\ &= -\sum_{i=1}^{n} p_i \cdot \log_b(q_i) \end{aligned}$$

This represents the average amount of surprise from observing an event when modelling a one distribution with another.

2.2.5 *Gibbs' inequality*

$$H(S||T) \geqslant H(S)$$

$$H(S||T) - H(S) \geqslant 0$$

The average amount of surprise when modelling one distribution with another is greater than if you had used the correct distribution.

2.2.6 *Relative Entropy*

This difference in entropies, $H(S||T) - H(S)$, is the relative entropy. Commonly known as Küllback-Liebler divegence; it is a measure of the divergence of information streams:

$$\begin{aligned} KL(S||T) &= H(S||T) - H(S) \\ &= -\sum_{i=1}^{n} p_i \cdot \log_b(q_i) + \sum_{i=1}^{n} p_i \cdot \log_b(p_i) \\ &= -\sum_{i=1}^{n} p_i \cdot (\log_b(q_i) - \log_b(p_i)) \\ &= -\sum_{i=1}^{n} p_i \cdot \log_b(\frac{q_i}{p_i}) \end{aligned}$$

The information in event e for discrimination between distributions P and Q, and the mean information for discrimination. As a general measure of difference, this function has a number of problems: it is asymmetric, unbounded, and undefined in the presence of zero values.

2.2.7 *Jensen-Shannon Divergence*

Lin [**?**] further examined the KL divergence and defined a modified function which shares the more desirable properties, and removes the undesirable ones.

Definition 13. *Consider two sources,* S *and* T*, emitting streams of events from the same set* $E = \{e_1, e_2, \ldots, e_n\}$*. Let* S *emit events with probability* $P = \{p_1, p_2, \ldots, p_n\}$ *and* T *emit events with probability* $Q = \{q_1, q_2, \ldots, q_n\}$*. Consider a notional stream* U *with probability distribution* $R = \{r_1, r_2, \ldots, r_n\}$ *for events in* E *given by the function:*

$$P_R(e_i) = \frac{p_i + q_i}{2}$$

The Jensen-Shannon divergence is the average of the relative entropies, $KL(S||U)$ *and* $KL(T||U)$.

$$\begin{aligned} JSD(S,T) &= \frac{KL(S||U) + KL(T||U)}{2} \\ &= \frac{-\sum_{i=1}^{n} p_i \cdot \log_b(\frac{r_i}{p_i}) - \sum_{i=1}^{n} q_i \cdot \log_b(\frac{r_i}{q_i})}{2} \\ &= \frac{1}{2}\sum_{i=1}^{n} p_i \cdot \log_b(\frac{p_i}{r_i}) + q_i \cdot \log_b(\frac{q_i}{r_i}) \end{aligned}$$

If base 2 logs are used, then the outcome is bounded in [0,1] with 0 meaning that the two distributions have equal probabilities for every event, and 2 meaning that for no event are both probabilities non-zero (see e.g. [**?**]). Despite being a simple derivation from Küllback-Liebler [**?**], the Jensen-Shannon divergence is positive, symmetric, bounded and well-defined in all circumstances, and also has the identity property.

Surprisingly recently, two authors [**?** **?**] have independently established that the Jensen-Shannon divergence is the square of a proper metric. Since then, the metric has attracted some more interest in both statistics and information theory, and deeper analysis [**?**] continues to show that the metric has some properties that, in short, should lend it to being, in many cases, the best possible semantic distance over any probabilistic generated form.

The fact that a form exists which is a proper metric immediately leads to the possibility of its use within metric indexing techniques. However any such excitement is short-lived in most cases; generative spaces are typically high-dimensional and sparse, and typical Intrinsic Dimensionality is very high – possibly in the thousands – excluding any possibility of successfully using metric indexing techniques.

2.2.8 *Kolmogorov Complexity*

Until now, we have discussed information in terms of random variables where entropy is average information. Another approach instead considers the actual information needed to specify strings of symbols. The Kolmogorov complexity, $K(X)$, is the minimal length of any input which leads to the output X, for some fixed universal

computer. The Kolmogorov complexity of the concatenation of two strings X and Y, K(XY), should be larger than K(X) but cannot be larger than the sum K(X) + K(Y)

Part II

SIMILARITY WITH INFORMATION THEORY

Information theory provides techniques for measuring the quantity of information contained within a given structure. In this part, we introduce a new universal distance metric which is based upon the relative information content between two structures. We show how this metric can be efficiently computed and how it compares to other algorithms in terms of relevance and accuracy.

3

INFORMATION DISTANCES

In this chapter, we explore the measurement of structural similarity in structured data. Since increasingly more information is stored in semi-structured data formats such as XML, it makes sense not only to measure the similarity of the textual content (as in traditional information retrieval) but of its structural meta-data too. This can be very useful when, for example, extracting document types, integrating heterogeneous data sources, merging data while cleaning it, or query by example.

3.1 COMPRESSION BASED DISTANCES

If Kolmogorov complexity measures the absolute information content of an object, a similarly defined measurement of information distance between two objects is the minimal information required to generate either from the other. This measure is universal, covering all other alternative informational distances as special cases, and machine-independent. This universality, however, prevents its computability.[]

The information distance between x and y is the length of the shortest program to transform x into y and y into x, and, up to a logarithmic additive term, is equal to the maximum of the conditional Kolmogorov complexities, which is the length of the shortest program with input y that outputs x[]. Because it is asymmetric, the conditional complexity $K(y|x)$ alone is an unsuitable information distance, thus the algorithmic informational distance between x and y is the sum of the relative complexities, $K(y|x) + K(x|y)$. This overestimates the information required to translate between x and y in case there is some redundancy between the information required to get from x to y and the information required to get from y to x.

3.1.1 *Normalised Compression Distance*

Bennet et al. introduced an information metric between data objects in [], in which they defined information distance (ID)

$$\mathrm{ID}(x, y) = \max\{K(x|y), K(y|x)\} \tag{1}$$

and its normalized version (NID):

$$\mathrm{NID}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{2}$$

They also showed it to have, despite minor infringements, metric properties.

Although, as we have discussed, Kolmogorov complexity is not computable, it can be viewed as the length of the best possible compression of x. The best compression, however, is also not computable, but it is approximated using standard compression algorithms. In [], Cilibrasi and Vitanyi defined a normalized compression distance (NCD), derived from NID, replacing the denominator $K(x)$ with $C(x)$, which is the length of the compressed x, and the numerator first to

$$\max\{K(x, y) - K(x), K(x, y) - K(y)\} \tag{3}$$

exploiting the additive property of Kolmogorov complexity [], then – as it is easier to handle concatenation with compression – to

$$\min\{C(xy), C(yx)\} - \min\{C(x), C(y)\} \tag{4}$$

The symmetry of many compression algorithms allows a further simplification to

$$C(xy) - \min\{C(x), C(y)\} \tag{5}$$

arriving at the definition of NCD found in []

$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \tag{6}$$

3.1.2 *Crossparsing Distance*

Ziv and Merhav[] showed that crossparsing two sources can be used to approximate the relative entropy (or KL-divergence) between them, in the same way that the Lempel-Ziv[] showed self-parsing (compression) approximates the entropy of a single source. In [], Helmer defined crossparsing distance (CPD) using a generalized definition of crossparsing to allow sequences of differing length.

They defined CPD by normalising and symmetrising as follows: First, using the codebook $s(x|y)$ – which is the multiset of phrases of x resulting from the crossparsing of x with respect to y – they normalize to $\frac{|s(x|y)\setminus y|}{|x|}$, by observing that there is at least one phrase if x is found as a substring of y and at most $|x|$ phrases, thus $1 \leqslant |s(x|y)| \leqslant |x|$. This gives the value 0 if $x = y$, and a maximum value of 1. Note that $s(x|y) \setminus y$ removes a single copy of y from the multiset $s(x|y)$, even if multiple copies exist. Secondly, a symmetric distance arises by taking the mean of normalised crossparsing both ways:

$$\mathrm{CPD}(x, y) = \frac{1}{2}\left(\frac{|s(x|y) \setminus y|}{|x|} + \frac{|s(y|x) \setminus x|}{|y|}\right) \tag{7}$$

Helmer found CPD performed better than the NCD at clustering and classification tasks, showing the limitation of compression distance: there is only a certain window when the difference between two data objects is measured. The Gzip algorithm first builds a dictionary for x; the difference between x and y is only measured when y is parsed, since at that point y is encoded using a code created for x. As the parsing of y continues, more of the encoding will rely on the parts of y that has already been scanned and less from the dictionary for x. Puglisi et al. [] interpret this as a process in which the compression algorithm learns how to encode y and unlearns how to do so for x. CPD, however, does not display this effect, since x is used to encode y

and vice versa, thus, avoiding the consideration of self-similarity when comparing objects.

The trouble with both CPD and NCD is that neither is a true metric. CPD does not satisfy triangle inequality, and while NCD satisfies triangle inequality up to an additive constant, this is not very pragmatic for similarity search.

3.2 STATISTICAL DISTANCES

Where Kolmogorov Complexity is the absolute information content of an object, entropy is its expected information content and a limit on its best possible compression[]. The compression distances above all rely on a string representation of the objects to compress and derive an approximate value for Kolmogorov Complexity; in the following, we consider the statistical properties of an object to derive its entropy directly. We calculate a form of structural complexity, which we use as the basis for a distance function. Avoiding Kolmogorov complexity in this way allows a computable function that does not rely on approximation.

3.2.1 *Structural entropic distance (SED)*

Recall that the Kolmogorov Complexity of a string is the size of the smallest program that generates that string. We require structural complexity to behave much the same way for structured objects: The joint complexity should equal the individual complexities when both structures are identical, and should equal the sum of the individual complexities when there is no common structure.

Consider a structured object as a conceptual generator of a stream that represents all possible navigation operations. Suppose an infinite process of random traversals gives the probability of a traversal; the collection of all traversals with associated probabilities is called an ensemble. Let X be the source emitting the stream of navigation

events $E = \{e_1, e_2, \ldots, e_n\}$ with associated probabilities $P = \{p_1, p_2, \ldots, p_n\}$. Assuming each traversal is independent the entropy of the stream is:

$$H_b(X) = -\sum_i p_i \log_b p_i \tag{8}$$

which is the average information of the events in the stream E. To eliminate the dependence on the logarithmic base we define the structural complexity

$$C(X) = b^{H_b(X)} \tag{9}$$

In Algorithmic information theory, joint complexity is the complexity of the concatenation of two strings[]; with structured data, however, this approach would be undesirable, since concatenation does not represent a merge. Instead we merge the streams event by event, taking half from each. Formally, let X and Y be the sources emitting the streams E and F; let Z be the notional source emitting the merged stream $G = \{g_1, g_2, \ldots, g_n\}$ where $E \subseteq G$ and $F \subseteq G$, then

$$P_Z(g_i) = \frac{P_X(g_i) + P_Y(g_i)}{2} \tag{10}$$

We define structural entropic distance, notionally, as the ratio of the joint complexity to the geometric mean of individual complexities, normalised into the range $[0, 1]$, which can now be defined concretely using structural complexity:

$$SED(X, Y) = \frac{C(Z)}{\sqrt{C(X) \cdot C(Y)}} - 1 \tag{11}$$

This distance is guided by the requirements of having a distance based on the amount of overlap of structural similarity.

3.2.2 *Relationship to Jensen-Shannon divergence*

The above distance function turns out to have a great deal of commonality with the Jensen-Shannon divergence (JSD), as shown by Theorem 4.1.3, and is a semantically

compelling metric; for any data type that can be characterised by an ensemble of event-probability pairs, the Jensen-Shannon similarity gives an estimate of the probability of that data's having been output by the same probabilistic generator.

Theorem 3.2.1. $\mathrm{SED}(x, y) = b^{\mathrm{JSD}(x,y)} - 1$

Proof.

$$\mathrm{SED}(X, Y) = \frac{C(Z)}{\sqrt{C(X) \cdot C(Y)}} - 1 \tag{12}$$

$$= \frac{b^{H_b(Z)}}{\sqrt{b^{H_b(X)} \cdot b^{H_b(Y)}}} - 1 \tag{13}$$

$$= \frac{b^{H_b(X)}}{b^{\frac{1}{2}(H_b(X)+H_b(Y))}} - 1 \tag{14}$$

$$= b^{H_b(Z)-\frac{1}{2}(H_b(X)+H_b(Y))} - 1 \tag{15}$$

Consider now only the exponent:

$$H_b(Z) - \frac{1}{2}(H_b(X) + H_b(Y)) \tag{16}$$

Substituting in the definition of entropy:

$$-\frac{1}{2} \sum_e 2 \cdot P_Z(e) \log_b P_Z(e) - P_X(e) \log_b P_X(e) - P_Y(e) \log_b P_Y(e) \tag{17}$$

and rewriting in the positive:

$$\frac{1}{2} \sum_e -2 \cdot P_Z(e) \log_b P_Z(e) + P_X(e) \log_b P_X(e) + P_Y(e) \log_b P_Y(e) \tag{18}$$

and substituting in the definition of $P_Z(e)$

$$\frac{1}{2} \sum_e -(P_X(e) + P_Y(e)) \log_b P_Z(e) + P_X(e) \log_b P_X(e) + P_Y(e) \log_b P_Y(e) \tag{19}$$

then simplifying

$$\frac{1}{2} \sum_e P_X(e) \cdot (\log_b P_X(e) - \log_b P_Z(e)) + P_Y(e) \cdot (\log_b P_Y(e) - \log_b P_Z(e)) \tag{20}$$

and combining logs gives:

$$\frac{1}{2} \sum_e P_X(e) \log_b \frac{P_X(e)}{P_Z(e)} + P_Y(e) \log_b \frac{P_Y(e)}{P_Z(e)} \tag{21}$$

which as shown in equation **??** is the Jensen-Shannon divergence, thus,

$$SED(X, Y) = b^{JSD(X,Y)} - 1 \tag{22}$$

□

In this context, the JSD is considered as an assessment of the difference of two independent probability distributions: that is, where a randomly sampled event has a number of different possible outcomes, each with its own assigned probability.

This probabilistic divergence model can be applied to many other data sets, essentially any set of data whose semantics can be usefully captured in a vector space model. For example, in Information Retrieval, the unigram generative model (see e.g. [**?** **?**]) has long been successfully used as a basis of similarity estimation among documents. In this model, a document is essentially reduced to a probabilistic generator, generating the terms within the document with probabilities in ratio to the number of appearances of each. Any such probabilistic metric can then be applied over these probability distributions to give a measurement of document similarity. We look at this further in chapter **??**. Many other contexts, for example the extraction of image features, can also be viewed as such a probabilistic generative process.

3.3 STRUCTURED DATA

Both compression-based and statistical distances require, for structural comparison, a representation of the structured object that excludes the content, leaving only the structural information. With compression based distances only a string representation is required, so a traversal that outputs the structural data is sufficient. Statistical distances, however, require a distribution of events to be constructed upon which a probabilistic model is built; this requires a series of random traversals, meanwhile keeping track of the frequency of structural events.

3.3.1 *String representations*

In his comparison of NCD and CPD, Helmer[] used four different methods to extract structural information from XML documents

TAGS Strip XML documents of all content (e.g. text nodes and attribute values) and extract all other nodes in document order by outputting their tag names (both opening and closing) and possible attribute names.

PAIRWISE Remove all content from the XML document and keep the remaining structural nodes. For each node, however, prepend the name of the parent node to its name. Again the document order is maintained but no closing tags are output.

FULL PATH After removing the content, prepend node names to the full path from the root node to the current node.

FAMILY ORDER Family Order traversal represents a compromise between breadth-first and depth-first traversal. All the children of a node are output en bloc. However, before doing so, all their descendants have to be processed.

3.3.2 *Statistical representation*

Extracting structural information for Statistical distances require a little more treatment; using the tree as a conceptual generator of events, an event stream is output by a random walk of the tree. These events collectively form an ensemble of event-probability pairs from which the entropy is calculated.

Consider the two trees in example 2, a stream of events is generated by a *random walk* markov process over the tree structure. A start node is selected at random, then at each step either an edge is traversed or the process is exited. Once the process is exited the resultant path forms an event in the probability space. An equivalence rela-

event e	$P_X(e)$	$P_Y(e)$	$P_Z(e)$
/People	$\frac{1}{33}$	$\frac{1}{23}$	$\frac{56}{759}$
/People/Person	$\frac{2}{33}$	$\frac{2}{23}$	$\frac{112}{759}$
/People/Person/Name	$\frac{2}{33}$	$\frac{2}{23}$	$\frac{112}{759}$
/People/Person/Age	$\frac{2}{33}$	0	$\frac{1}{33}$
/People/Person/D.O.B	0	$\frac{2}{23}$	$\frac{1}{23}$
/People/Person/Organisation	0	$\frac{2}{23}$	$\frac{1}{23}$
/People/Person/Name/Firstname	$\frac{2}{33}$	0	$\frac{1}{33}$
/People/Person/Name/Surname	$\frac{2}{33}$	0	$\frac{1}{33}$
/Person	$\frac{2}{33}$	$\frac{2}{23}$	$\frac{112}{759}$
/Person/Name	$\frac{2}{33}$	$\frac{2}{23}$	$\frac{112}{759}$
/Person/Age	$\frac{2}{33}$	0	$\frac{1}{33}$
/Person/D.O.B	0	$\frac{2}{23}$	$\frac{1}{23}$
/Person/Organisation	0	$\frac{2}{23}$	$\frac{1}{23}$
/Person/Name/Firstname	$\frac{2}{33}$	0	$\frac{1}{33}$
/Person/Name/Surname	$\frac{2}{33}$	0	$\frac{1}{33}$
/Name	$\frac{2}{33}$	$\frac{2}{23}$	$\frac{112}{759}$
/Name/Firstname	$\frac{2}{33}$	0	$\frac{1}{33}$
/Name/Surname	$\frac{2}{33}$	0	$\frac{1}{33}$
/Age	$\frac{2}{33}$	0	$\frac{1}{33}$
/Firstname	$\frac{2}{33}$	0	$\frac{1}{33}$
/Surname	$\frac{2}{33}$	0	$\frac{1}{33}$
/D.O.B	0	$\frac{2}{23}$	$\frac{1}{23}$
/Organisation	0	$\frac{2}{23}$	$\frac{1}{23}$

Table 1: all of the events generated by random walks and their probabilities

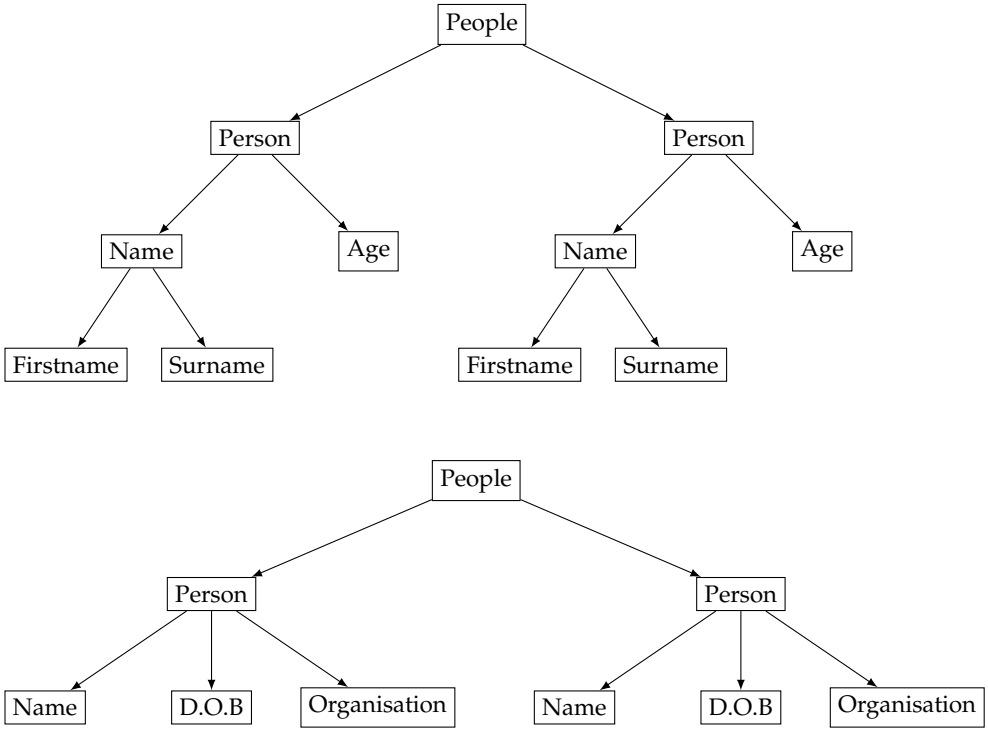


Figure 2: Trees used to generate an ensemble of events

tion is formed on paths that contain a sequence of nodes with identical labels, that is, two different paths may be considered an instance of the same event if their sequence of node labels are identical. We can then calculate the probabilities of the events by simply enumerating all possible paths in the tree and counting the occurrences of each event. Table 1 shows the probabilities of each event in both trees, along with the probabilities of the merged stream.

3.3.3 *Mapping to a vector space*

Many important similarity applications make use of the vector space model, where data objects can be represented as points in a space (for more information consult chapter **??**). The compression based distances do not lend themselves to this kind of treatment, not least because they do not satisfy Triangle Inequality. The statistical distances do, however.

Using this mapping, tree structured data can easily be adapted to existing similarity applications. We can, furthermore, now treat trees simply as vectors in a vector space and benefit from the abstraction knowing that any mathematical treatment applies to the whole domain, and indeed any other. We now show the mapping from a frequency table of events to a vector space, and the calculation of JSD for vectors in $\mathbb{R}^n$.

Consider an inner product space over $\mathbb{R}^n$, where each dimension x_i corresponds to a count of event e_i. We use the norm:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{23}$$

to produce the unit vector:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_1} \tag{24}$$

where each $\hat{x}_i$ corresponds to the probability of observing event e_i. Now, let the information vector of $\mathbf{x}$ be:

$$\mathbf{i}_x = -\log_b(\hat{\mathbf{x}}) \tag{25}$$

where each i_j is the amount of information gained by observing event e_j. We can use the inner product:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i \tag{26}$$

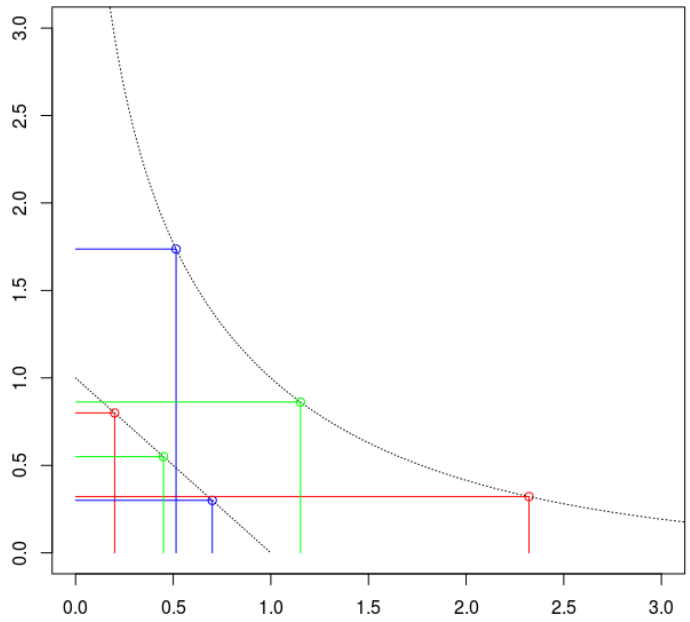


Figure 3: In two dimensions – The unit vectors $\hat{\mathbf{x}}$ (red), $\hat{\mathbf{y}}$ (blue), and $\hat{\mathbf{z}} = \frac{\hat{\mathbf{x}}+\hat{\mathbf{y}}}{2}$ (green) are on the L^1 norm; and the information vectors (using base 2), $\mathbf{i}_x$, $\mathbf{i}_y$, $\mathbf{i}_z$ are on the curved line. The entropy is the dot product of the unit vector and information vector. As the unit vector moves to the extremes in one direction the

to calculate the entropy as the inner product of the unit vector $\hat{\mathbf{x}}$ and the information vector $\mathbf{i}_x$:

$$H_b(\mathbf{x}) = \langle \hat{\mathbf{x}}, \mathbf{i}_x \rangle \tag{27}$$

which is the total information in the vector $\mathbf{x}$. And finally calculate the Jensen-Shannon divergence as,

$$\mathrm{JSD}(\mathbf{x}, \mathbf{y}) = H_b(\mathbf{z}) - \frac{1}{2}(H_b(\mathbf{x}) + H_b(\mathbf{y})) \tag{28}$$

where $\mathbf{z} = \frac{\hat{\mathbf{x}}+\hat{\mathbf{y}}}{2}$ is the centroid of the unit vectors.

Plotting some of these vectors in two-dimensional spaces gives us some intuition for how this function behaves. As figure 3 demonstrates, the highest entropies occur

when the normalised points lie at the extreme ends of the norm; by implication distances are far larger at the extremes than its Euclidean representation might suggest. When a point has one dimension that is very low, it pushes that point much further away from other points that do not. And this is, in fact, desirable behaviour for many similarity applications, such as clustering.

3.4 CONCLUSION

This chapter has shown that existing information theoretic measures of structural distance based upon compression may not be suitable for similarity search. We have produced an alternative, SED, that uses entropy at its core, which in turn has many commonalities with Jensen-Shannon Divergence.

The transformation of JSD for use with metric searching prohibitively increases its dimensionality. We have shown that SED, however, which can be used in its raw form for tree structured data, allows for much greater pruning opportunities with a metric index. Furthermore, through algebraic manipulation, it can be calculated efficiently; by using an inverted index data structure, and, for range queries, early termination, this saving is as much as two orders of magnitude.

4

MULTIWAY STRUCTURAL ENTROPIC DISTANCE

Much of the research on similarity search focuses on the similarity (or distance) between two objects. For many situations, however, ascertaining the mutual similarity of a set of objects would be useful. Applications could be, for example, similarity joins, clustering, cluster analysis, and potentially many more that are dependent upon these techniques.

The calculation of density, or multi-way divergence as the analogue to distance, has been rarely used and is typically calculated through some compound function over the set of pair-wise distances within the set of objects: for example, the mean inter-set distance or the mean distance from each object to a centroid.

In this chapter, we derive a function to calculate the divergence of a set of objects that is based on the structural entropic distance discussed in chapter **??**. This new metric, which is grounded in information theory, avoids the problem of repeated calls to the distance metric through a direct, calculable notion of multi-way divergence without relying on approximation. It reuses the notion of complexity offered in the definition of the original distance metric and reverts to this definition when the size of the set is two. It is bounded, giving a maximum value when there is no commonality, allowing absolute comparisons to be made. And finally, it is only a little more expensive than a single distance to evaluate.

We show that multi-way divergence offers a cheaper alternative to compound functions whilst giving similar, or better, semantic properties. We believe it could be applied usefully to construct new metric indices or to execute more complex queries, such as similarity joins, efficiently.

4.1 MULTIWAY DISTANCES

The notion of a multi-way distance metric is not new, motivation coming from geometry and topology. Recently a few papers have analysed the generalisation to multi-way for any existing metric[**? ? ?**]. They consider whether various axioms are observed in these generalisations. For example, a metric space with a simple dyadic metric has the following axioms:

$$d(x, y) = 0 \Leftrightarrow x = y$$

$$d(x, y) = d(y, x)$$

$$d(x, y) \leqslant d(x, z) + d(y, z)$$

The first of these, is simply generalised to $D(x_1, \ldots, x_n) = 0$ if and only if all x_i are equal; the second to $D(x_{\pi(1)}, \ldots, x_{\pi(n)}) = D(x_1, \ldots, x_n)$ for every permutation π of $\{1, 2, \ldots, n\}$; while the third, triangle inequality, has been generalised in numerous ways[**?**], such as polyhedron inequality:

$$(n-1) \cdot D(x_1, \ldots, x_n) \leqslant \sum_{i=1}^{n} D(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n+1})$$

These generalised axioms are interesting in their own right, and the first two are desirable properties of a multi-way divergence function, but it is not clear if polyhedron inequality aids similarity search at this stage.

4.1.1 *Compound metrics*

Deza and Rosenberg introduced the multi-way extension of the star distance in [**?**], while perimeter distance[**?**] gives a geometrical "average distance". Both of these, however, are compound functions and do not fundamentally look at generalising any specific metric.

4.1.2 *Multiway structural entropic distance (MSED)*

Let $X_1, \ldots X_n$ be the sources emitting the streams $E_1, \ldots E_n$; let Y be the notional source emitting the merged stream $F = \{f_1, f_2, \ldots, f_m\}$ where $\forall E_i, E_i \subseteq F$, then

$$P_Y(f_i) = \frac{\sum_{j=1}^{n} P_{X_j}(f_i)}{n} \tag{29}$$

Recall from chapter **??** that the complexity of a source X_i is given by the logarithmic base b raised to the power of the entropy $H_b(X_i) = -\sum_{e \in E_i} P_{X_i}(e) \log_b P_{X_i}(e)$ of that source, $C(X_i) = b^{H_b(X_i)}$.

Using the same principle as before of applying the ratio of the complexity of the merged stream to the geometric mean of individual complexities, we arrive at the ratio

$$\frac{C(Y)}{\sqrt[n]{\prod_{i=1}^{n} C(X_i)}} \tag{30}$$

which gives a value in the range $[1, n]$; we scale this to $[0, 1]$ to give the multiway structural entropic distance:

$$MSED(X_1, \ldots X_n) = \frac{1}{n-1} \cdot \left(\frac{C(Y)}{\sqrt[n]{\prod_{i=1}^{n} C(X_i)}} - 1 \right) \tag{31}$$

As it should, the joint complexity equals the individual complexities when all structures are identical, and equals the sum of the individual complexities when there is no common structure.

Using the same mapping to vector spaces as before, this generalised function remains a ratio of the complexity of a centroid to the geometric mean of its neighbours' complexities, and has the following properties:

- If all vectors are identical it gives 0
- If all vectors are different it gives 1
- All other inputs give a value between these bounds

Consider now the metric space axioms described at the beginning of this section; we have stated that the lower bound is achieved when all elements are the same, so the first axiom holds. Since all operations involved in both the numerator and denominator are associative, total symmetry holds too. We do not yet know whether polyhedron inequality holds.

4.1.3 *Relationship to Jensen-Shannon divergence*

As we saw in the last chapter, SED is related to Jensen-Shannon. A similar relationship exists between the two multi-way versions. Recall that Jensen-Shannon may be expressed in vector notation as:

$$\mathrm{JSD}(\mathbf{x}, \mathbf{y}) = \mathrm{H}_b(\frac{1}{2}(\mathbf{x} + \mathbf{y})) - \frac{1}{2}(\mathrm{H}_b(\mathbf{x}) + \mathrm{H}_b(\mathbf{y})) \tag{32}$$

When defining Jensen-Shannon divergence, Lin[] also described a multi-way generalisation:

$$\mathrm{JSD}(\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(n)}}) = \mathrm{H}_b(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x^{(i)}}) - \frac{1}{n}\sum_{i=1}^{n}\mathrm{H}_b(\mathbf{x^{(i)}}) \tag{33}$$

Theorem 4.1.1. $\mathrm{MSED}(\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(n)}}) = \frac{b^{\mathrm{JSD}(\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(n)}})} - 1}{n-1}$

Proof.

$$\mathrm{MSED}(\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(n)}) = \frac{1}{n-1}\cdot\left(\frac{C(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)})}{\sqrt[n]{\prod_{i=1}^{n}C(\mathbf{x}^{(i)})}}-1\right) \tag{34}$$

$$= \frac{1}{n-1}\cdot\left(\frac{b^{H_b(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)})}}{\sqrt[n]{\prod_{i=1}^{n}b^{H_b(\mathbf{x}^{(i)})}}}-1\right) \tag{35}$$

$$= \frac{1}{n-1}\cdot\left(\frac{b^{H_b(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)})}}{b^{\frac{1}{n}\sum_{i=1}^{n}H_b(\mathbf{x}^{(i)})}}-1\right) \tag{36}$$

$$= \frac{1}{n-1}\cdot(b^{H_b(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)})-\frac{1}{n}\sum_{i=1}^{n}H_b(\mathbf{x}^{(i)})}-1) \tag{37}$$

$$= \frac{b^{JSD(\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(n)})}-1}{n-1} \tag{38}$$

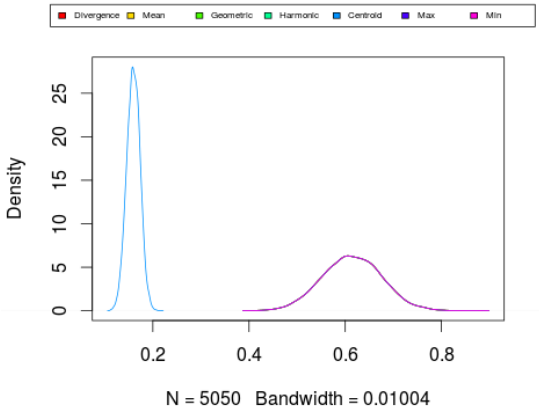
□

4.2 CORRELATIONS

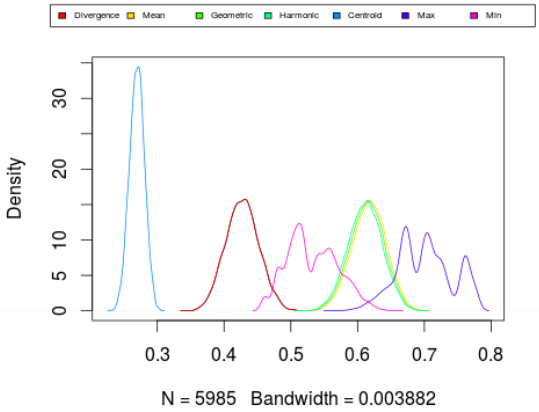
The best correlation comes with the mean distance to a centroid. This method is calculated by making a centroid using the mean vector then averaging all distances in the cluster to it. Since multi-way divergence is a ratio of the complexity of a centroid to the geometric mean of individual complexities, there is much more in common with this definition. Even when the comparison distance metric is Euclidean distance, a very strong correlation exists (figure 7c).

The mean intra-cluster distance also correlates well with multi-way divergence. Figure 7a shows the correlation with the mean intra-cluster distance, which appears to be strongest at the, more commonly used, lower end.

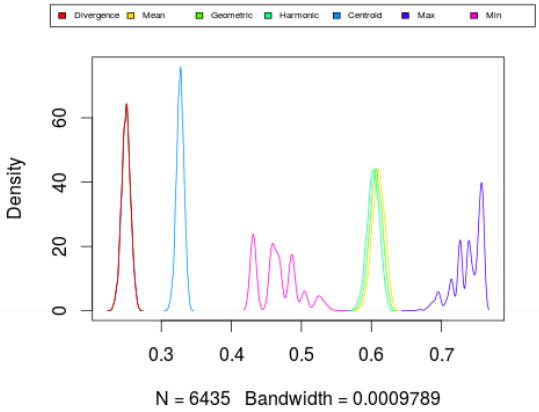
Multi-way divergence correlates – less strongly than the other methods – with both SED and Euclidean maximum intra-cluster distance, and the correlation drops as the cluster size increases. Rather than assessing the mutual similarity, the maximum distance simply describes the two farthest points in the cluster. These two points must



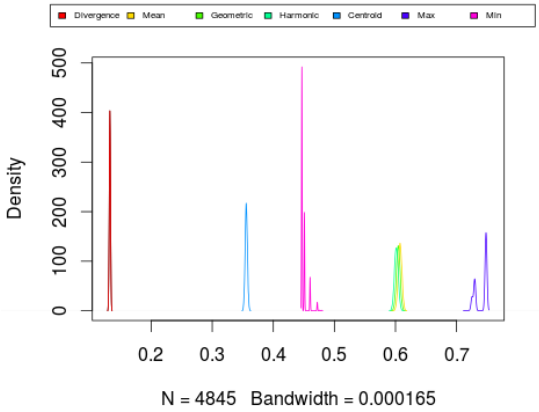
(a) triples with mean structural entropic distance



(b) 5-tuples with mean euclidean distance to a centroid



(c) 5-tuples with mean euclidean distance to a centroid

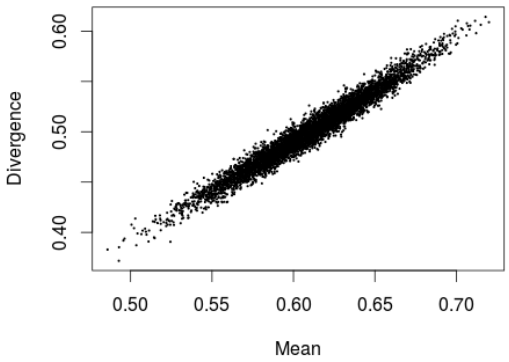


(d) 5-tuples with mean euclidean distance to a centroid

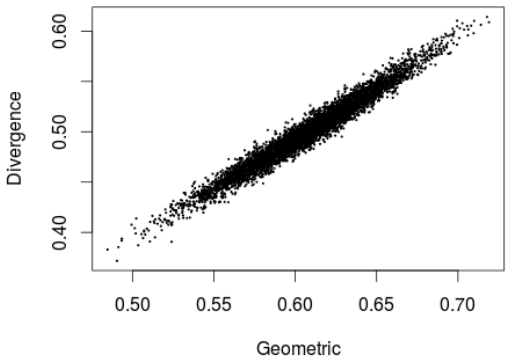
Figure 4: Correlation with divergence in 15-dimensional space

lie on the cluster perimeter and describe the spread of points across the space. Using only two points, however, fails to account for the spread in other dimensions, further verified by the drop in correlation in the higher dimensional space.

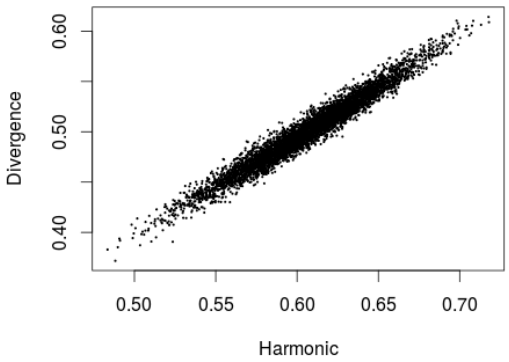
4.3 SUMMARY



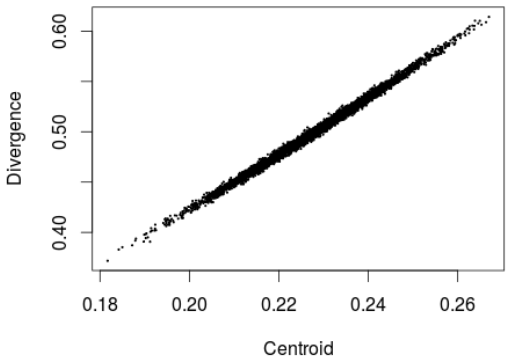
(a) triples with mean structural entropic distance



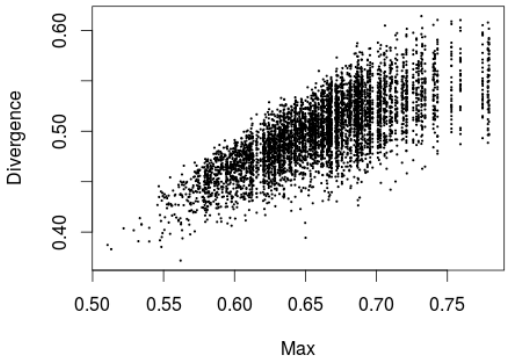
(b) 5-tuples with mean euclidean distance to a centroid



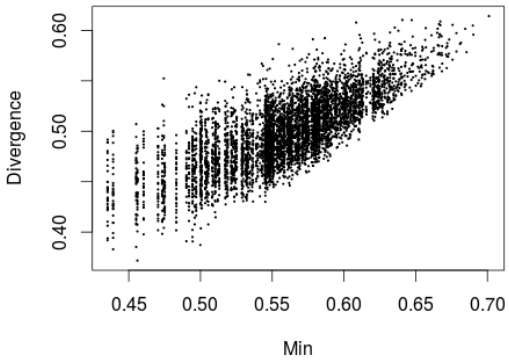
(c) 5-tuples with mean euclidean distance to a centroid



(d) 5-tuples with mean euclidean distance to a centroid

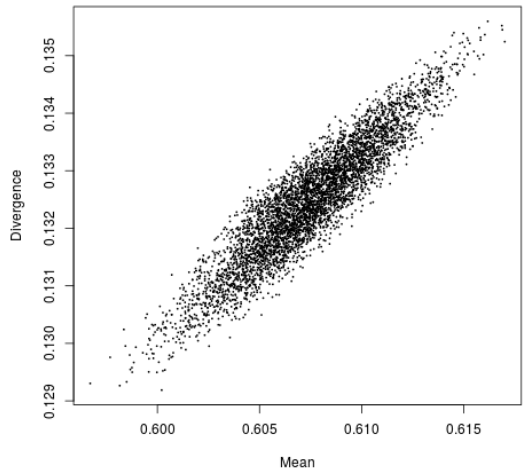


(e) 5-tuples with mean euclidean distance to a centroid

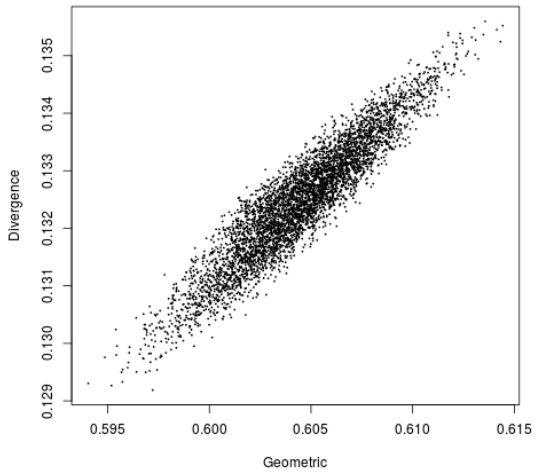


(f) 5-tuples with mean euclidean distance to a centroid

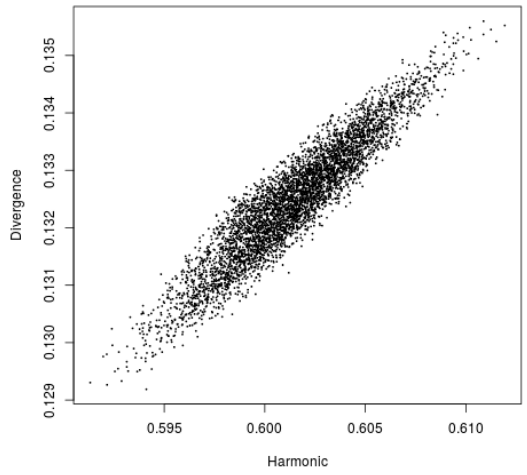
Figure 5: Correlation with divergence in 15-dimensional space



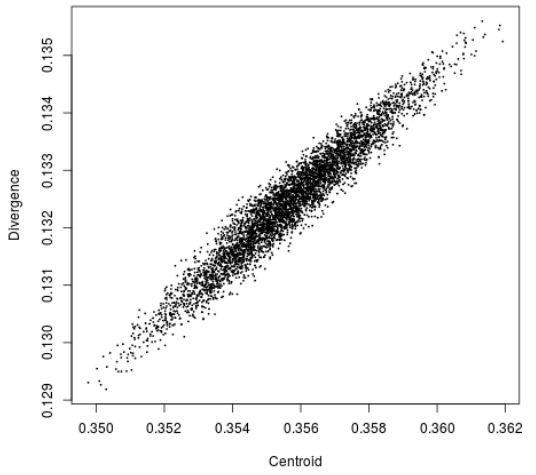
(a) triples with mean structural entropic distance



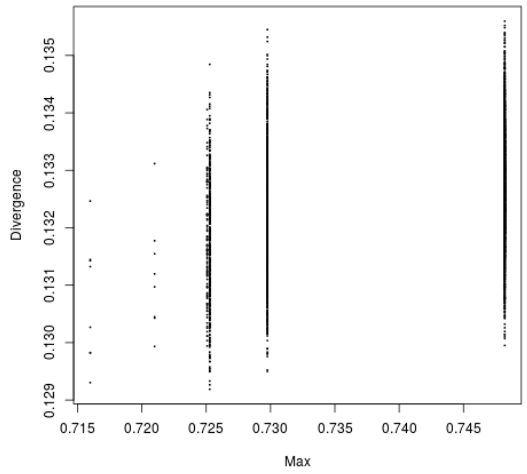
(b) 5-tuples with mean euclidean distance to a centroid



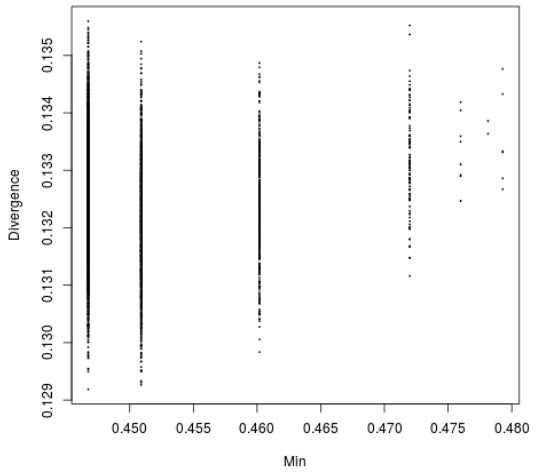
(c) 5-tuples with mean euclidean distance to a centroid



(d) 5-tuples with mean euclidean distance to a centroid

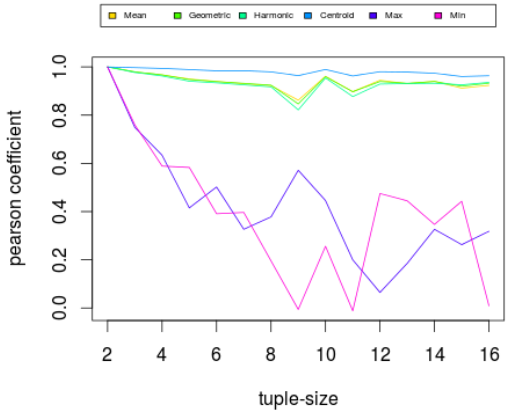


(e) 5-tuples with mean euclidean distance to a centroid

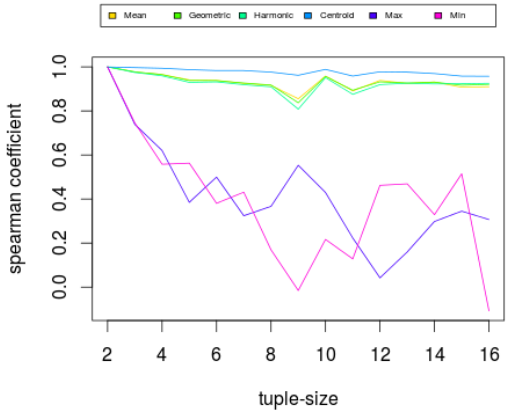


(f) 5-tuples with mean euclidean distance to a centroid

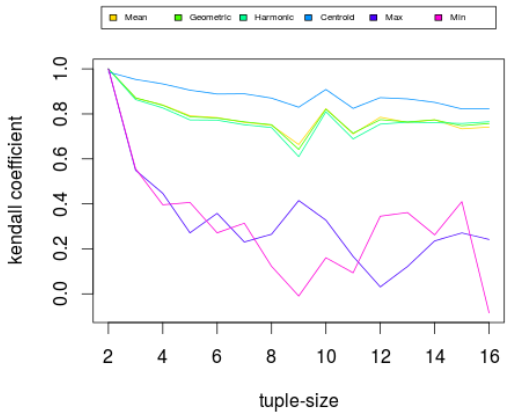
Figure 6: Correlation with divergence in 15-dimensional space



(a) triples with mean structural entropic distance



(b) 5-tuples with mean euclidean distance to a centroid



(c) 5-tuples with mean euclidean distance to a centroid

Figure 7: Correlation with divergence in 15-dimensional space

5

EFFICIENT EVALUATION

In this chapter, we derive an equivalent metric first with the standard two-way SED using only intersecting dimensions to significantly reduce its cost, and secondly, generalising to the multi-way MSED whereby if there are n input vectors we only need to consider dimensions that are non-zero in > 1 of the input vectors, avoiding the cost of calculating logs.

We evaluate over both sparse vectors and inverted indices which allow parallel evaluation over large data spaces. We show that a lower bound can be efficiently maintained during calculation so that the calculation may be eagerly abandoned.

5.1 AVOIDING CALCULATION OF THE MEAN

We consider the efficient evaluation of JSD, since it can be used in the calculation of MSED. In particular, we are interested in an incremental evaluation since evaluation over large collections can be achieved through the use of an inverted index data structure, saving space.

5.1.1 *2-way case*

Although it appears that JSD requires the construction of a new vector $\mathbf{z}$, in fact, $\mathbf{z}$ does not need to be constructed. Working with the vector notation from section 3.3.3 we can see this as follows

$$\mathrm{JSD}(\mathbf{x}, \mathbf{y}) = H_b(\mathbf{z}) - \frac{1}{2}(H_b(\mathbf{x}) + H_b(\mathbf{y})) \tag{39}$$

$$= -\sum_i z_i \log_b z_i - \frac{1}{2}(-\sum_i x_i \log_b x_i - \sum_i y_i \log_b y_i) \tag{40}$$

$$= -\frac{1}{2}\sum_i 2 \cdot z_i \log_b z_i - (x_i \log_b x_i + y_i \log_b y_i) \tag{41}$$

$$= \frac{1}{2}\sum_i x_i \log_b x_i + y_i \log_b y_i - (x_i + y_i) \log_b \frac{(x_i + y_i)}{2} \tag{42}$$

avoiding the construction of $\mathbf{z}$.

Using base 2 logs allows the elimination of yet more terms:

$$\mathrm{JSD}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\sum_i x_i \log_2 x_i + y_i \log_2 y_i - (x_i + y_i)(\log_2(x_i + y_i) - 1) \tag{43}$$

$$= \frac{1}{2}\sum_i x_i \log_2 y_i + x_i \log_2 y_i - (x_i + y_i) \log_2(x_i + y_i) + x_i + y_i \tag{44}$$

and, since $\sum_i x_i = \sum_i y_i = 1$,

$$\mathrm{JSD}(\mathbf{x}, \mathbf{y}) = 1 + \frac{1}{2}\sum_i x_i \log_2 x_i + y_i \log_2 y_i - (x_i + y_i) \log_2(x_i + y_i) \tag{45}$$

Since we are interested in the incremental evaluation, we negate the summand function and name it:

$$\mathcal{F}(x, y) = (x + y) \log_2(x + y) - x \log_2 x - y \log_2 y \tag{46}$$

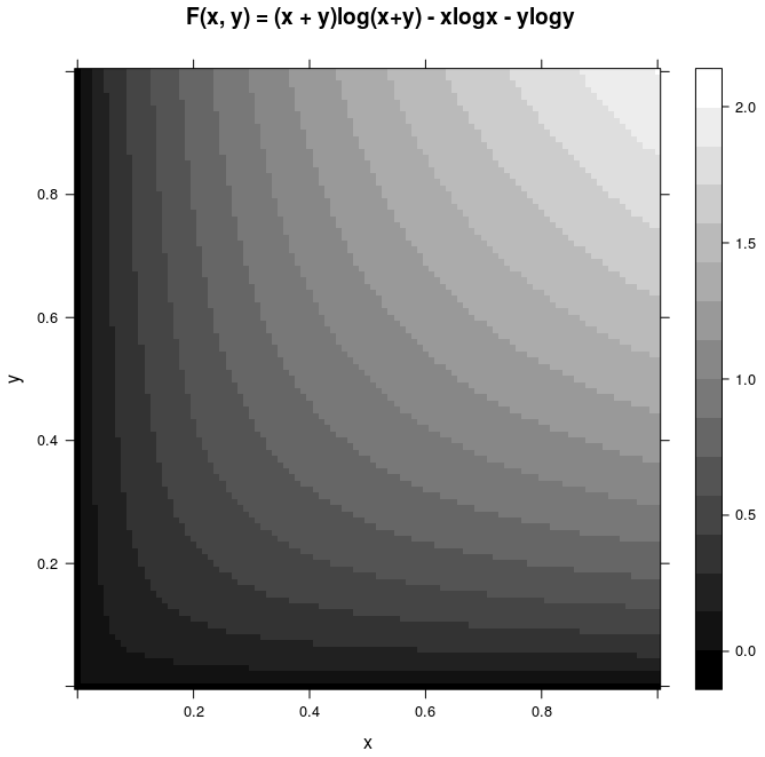


Figure 8: The function $\mathcal{F}$ on all possible inputs

giving:

$$\mathrm{JSD}(\mathbf{x}, \mathbf{y}) = 1 - \frac{1}{2} \sum_i \mathcal{F}(x_i, y_i) \tag{47}$$

where if either (or both) of x_i or y_i is equal to zero then so is $\mathcal{F}(x_i, y_i)$, thus only the non-zero intersection is required; that is, $\mathcal{F}(x_i, y_i)$ need only be evaluated when $x_i \neq 0$ and $y_i \neq 0$.

5.1.2 *multi-way case*

In the multi-way case we follow the same reasoning. Starting with the definition of JSD we avoid the construction of $\mathbf{z}$ as follows:

$$\mathrm{JSD}(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = H_b(\mathbf{z}) - \frac{1}{n}(\sum_j H_b(\mathbf{x}^{(j)})) \tag{48}$$

$$= -\sum_i z_i \log_b z_i - \frac{1}{n}(\sum_j -\sum_i x_i^{(j)} \log_b x_i^{(j)}) \tag{49}$$

$$= -\frac{1}{n}\sum_i n \cdot z_i \log_b z_i - (\sum_j x_i^{(j)} \log_b x_i^{(j)}) \tag{50}$$

$$= \frac{1}{n}\sum_i (\sum_j x_i^{(j)} \log_b x_i^{(j)}) - (\sum_j x_i^{(j)}) \log_b \frac{(\sum_j x_i^{(j)})}{n} \tag{51}$$

This time using base n logs allows the elimination of extra terms:

$$\mathrm{JSD}(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = \frac{1}{n}\sum_i (\sum_j x_i^{(j)} \log_n x_i^{(j)}) - (\sum_j x_i^{(j)})(\log_n \sum_j x_i^{(j)} - 1) \tag{52}$$

$$= \frac{1}{n}\sum_i (\sum_j x_i^{(j)} \log_n x_i^{(j)}) - (\sum_j x_i^{(j)}) \log_n (\sum_j x_i^{(j)}) + (\sum_j x_i^{(j)}) \tag{53}$$

and, since $\sum_i x_i^{(j)} = \sum_i x_i^{(k)} = 1$, for any j, k

$$\mathrm{JSD}(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = 1 + \frac{1}{n}\sum_i (\sum_j x_i^{(j)} \log_n x_i^{(j)}) - (\sum_j x_i^{(j)}) \log_n (\sum_j x_i^{(j)}) \tag{54}$$

again, we are interested in the incremental evaluation, so we generalise the $\mathcal{F}$ function from the 2-way case:

$$\mathcal{F}(x^{(1)}, \ldots, x^{(n)}) = (\sum_j x^{(j)}) \log_n (\sum_j x^{(j)}) - (\sum_j x^{(j)} \log_n x^{(j)}) \tag{55}$$

giving:

$$\mathrm{JSD}(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = 1 - \frac{1}{n} \sum_i \mathcal{F}(x_i^{(1)}, \ldots, x_i^{(n)}) \tag{56}$$

As before, we notice that $\mathcal{F}$ is equal to zero when $n-1$ or more of the n input values is zero. Thus, we can avoid the expensive calculation of the logs and replace it with the cheaper count of zeros.

5.2 EARLY TERMINATION

The outcome of the metric, with range queries in similarity search, is of no interest if it is greater than the threshold r. We now show a way of optimising the calculation further.

The threshold requirement is written

$$\frac{n^{1-\frac{1}{n}\sum_{i=1}^{n} \mathcal{F}(x_i^{(1)}, \ldots, x_i^{(n)})} - 1}{n-1} \leqslant r \tag{57}$$

$$\Rightarrow \quad n^{1-\frac{1}{n}\sum_{i=1}^{n} \mathcal{F}(x_i^{(1)}, \ldots, x_i^{(n)})} - 1 \leqslant (n-1)r \tag{58}$$

$$\Rightarrow \quad 1 - \frac{1}{n} \sum_{i=1}^{n} \mathcal{F}(x_i^{(1)}, \ldots, x_i^{(n)}) \leqslant \log_n((n-1)r + 1) \tag{59}$$

$$\Rightarrow \quad -\frac{1}{n} \sum_{i=1}^{n} \mathcal{F}(x_i^{(1)}, \ldots, x_i^{(n)}) \leqslant \log_n((n-1)r + 1) - 1 \tag{60}$$

$$\Rightarrow \quad \sum_{i=1}^{n} \mathcal{F}(x_i^{(1)}, \ldots, x_i^{(n)}) \geqslant n - n\log_n((n-1)r + 1) \tag{61}$$

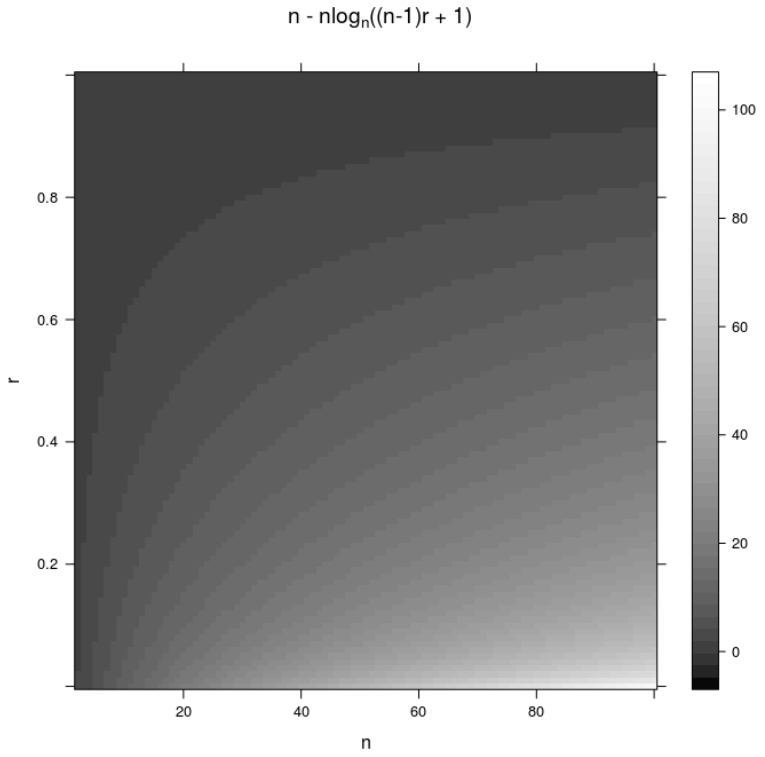


Figure 9: Similarity lower bounds

$\mathcal{F}$ can be seen as a similarity accumulator, and the term $n \log_n((n-1)r+1)$ as the maximum shortfall that may occur in order for the threshold r not to be exceeded.

If, at any point of the iterative calculation, we can determine that it is impossible for the value of $\sum_{i=1}^{n} \mathcal{F}(x^{(i)}, \ldots, x^{(n)})$ to reach the threshold of $n - n \log_n((n-1)r+1)$, then the calculation may be abandoned.

We calculate an upper bound for $\sum_{i=1}^{n} \mathcal{F}(x^{(i)}, \ldots, x^{(n)})$ after k stages of the iteration by consideration of the sum of terms:

$$\sum_{i=1}^{n} \mathcal{F}(x^{(i)}, \ldots, x^{(n)}) = \sum_{i=1}^{k} \mathcal{F}(x^{(i)}, \ldots, x^{(n)}) + \sum_{i=k+1}^{n} \mathcal{F}(x^{(i)}, \ldots, x^{(n)}) \tag{62}$$

At stage k, the value of the left hand term is known, and using the Jensen inequality we can calculate an upper bound for the right-hand term:

$$\sum_{i=k+1}^{n} \mathcal{F}(x^{(i)}, \ldots, x^{(n)}) \quad \leqslant \quad \mathcal{F}\left(\sum_{i=k+1}^{n} x_i^{(1)}, \ldots, \sum_{i=k+1}^{n} x_i^{(n)} \right) \tag{63}$$

$$\Rightarrow \quad \sum_{i=k+1}^{n} \mathcal{F}(x^{(i)}, \ldots, x^{(n)}) \quad \leqslant \quad \mathcal{F}\left(1 - \sum_{i=1}^{k} x_i^{(1)}, \ldots, 1 - \sum_{i=1}^{k} x_i^{(n)} \right) \tag{64}$$

without knowledge of even the number of dimensions still to be computed.

Thus if at any stage k the inequality

$$\sum_{i=1}^{k} \mathcal{F}(x^{(i)}, \ldots, x^{(n)}) + \mathcal{F}\left(1 - \sum_{i=1}^{k} x_i^{(1)}, \ldots, 1 - \sum_{i=1}^{k} x_i^{(n)} \right) < n - n \log_n((n-1)r + 1) \tag{65}$$

becomes true the calculation may be terminated early with the knowledge that the distance falls out with the range of the query.

Whether the optimisation based on this observation is worth applying or not depends entirely on the context of the calculation; it requires the calculation of an extra application of $\mathcal{F}$ and, if applied at each stage of the calculation, would only be cost-effective if a saving equivalent to half the number of stages was achieved. This is not to say that the observation is of no use, and in particular there may be circumstances where an I/O saving can be achieved.

5.3 EVALUATION

Based on the above algebraic observations, we implemented five different methods of evaluating the metric, as follows. All of which avoided the calculation of the mean vector:

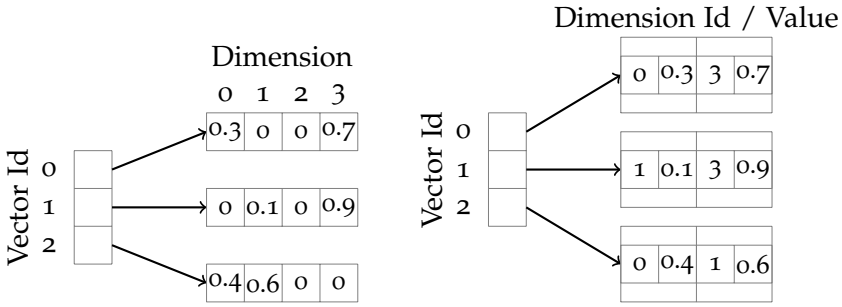
INVERTED INDEX — Using an inverted index (skipping dimensions where > n-1 values are 0);

INVERTED INDEX ET — using an inverted index with early termination (skipping dimensions where > n-1 values are 0).

SPARSE VECTORS — All dimensions of the vectors being compared are accessed;

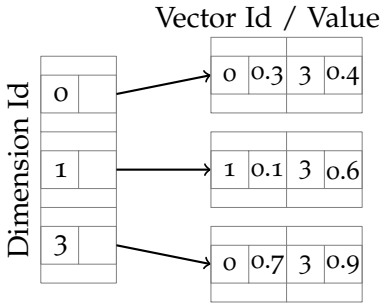
SPARSE VECTORS 2 — skipping dimensions where > n-1 values are 0;

SPARSE VECTORS 3 — skipping dimensions where > n-1 values are 0, with early termination;



(a) Simple Vector Representation

(b) Sparse Vector Representation

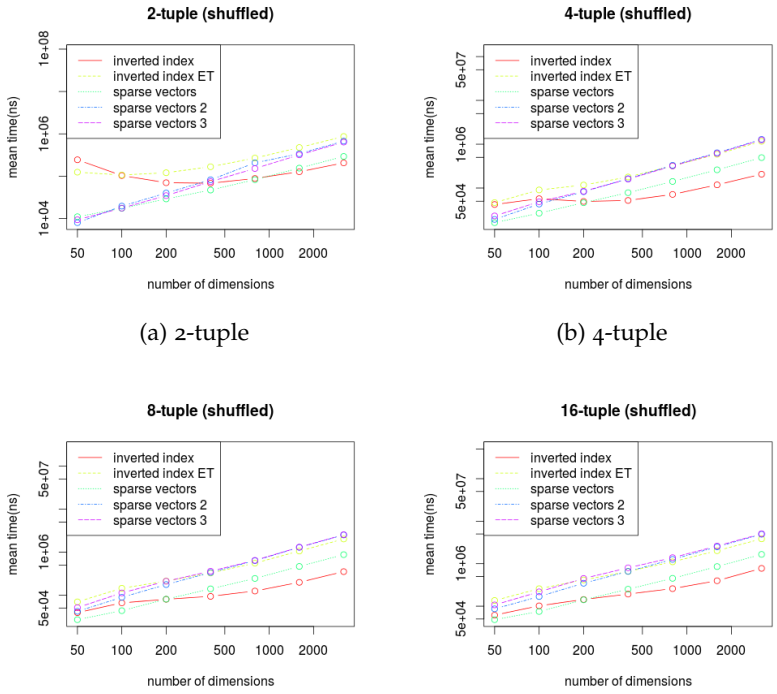


(c) Inverted Index Representation

Figure 10

A number of generated spaces were used to test the mechanisms. The generator was set to populate sparse spaces of 50, 100, 200, 400, 800, 1600 and 3200 dimensions: 50 dimensions being populated in each. We used two types of generator, which produced what we call either shuffled or unshuffled vectors. In the unshuffled vectors, the generator populated the first 50 dimensions of each vector; in the shuffled vectors, the generator evenly distributed the populated dimensions among the n dimensions by randomly shuffling the unshuffled vectors' dimensions. Search thresholds to return 10^{-4} of the data were then calculated for each space.

A metric index structure achieves no significant cost saving in these spaces; the space with 50 dense dimensions has an IDIM of x, with median distance of x. IDIM



(a) 2-tuple

(b) 4-tuple

(c) 8-tuple

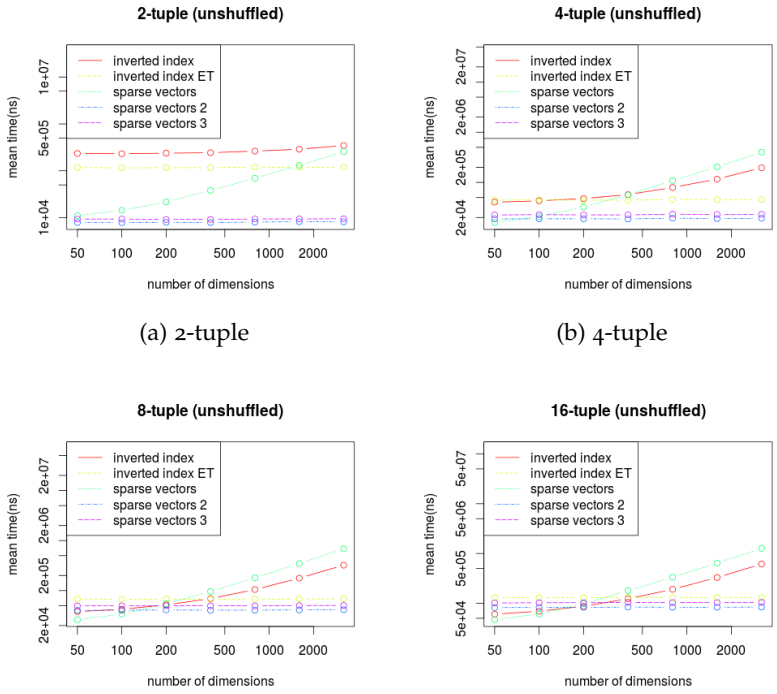
(d) 16-tuple

Figure 11: Comparison of evaluation techniques on the shuffled datasets over increasing sparsity

increases, furthermore, with both the number of dimensions in the space and the relative sparsity.

Figure 11 shows the effect of increasing the sparsity on shuffled vectors. Initially, the sparse vectors are much quicker than the inverted index, but as the sparsity increases the inverted index becomes more efficient than the sparse vectors; this gap narrows, however, as the tuple size increases. In all cases, the cost of maintaining the threshold invariant outweighs the saving from early termination.

Figure 12 shows the effect of increasing the sparsity on unshuffled vectors. With the 2-tuples shown in 12a, an inverted index is slower than the sparse vector representations, but benefits from using early termination. The sparse vectors, also however, benefit significantly both from skipping dimensions and early termination. As the tuple size is increased, the difference between sparse vectors and the inverted index narrows, and the inverted index even runs more efficiently beyond a certain point of sparsity. Again, the higher tuple size show an impressive saving from early termination both in the inverted index and sparse vectors.



(a) 2-tuple (b) 4-tuple

(c) 8-tuple (d) 16-tuple

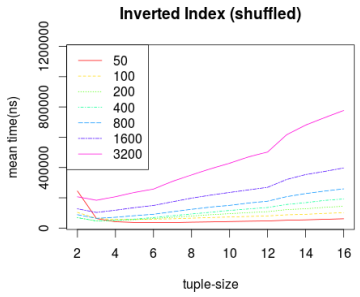
Figure 12: Comparison of evaluation techniques on the unshuffled datasets over increasing sparsity

Figure 13 shows the growth of each mechanism on the shuffled data. The growth of pattern is the same for all mechanisms: dense spaces show no extra cost when increasing the tuple size, but as the sparsity increases so does the rate of growth.

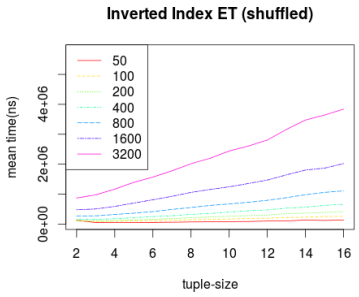
Figure 14 shows the growth of each mechanism on the unshuffled data. The effect of early termination makes all levels of sparsity behave the same. Whereas without early termination, the growth is larger for more sparse vectors.

Figure 15 compares the growth associated with increasing tuple size of each mechanism on the shuffled data. On all levels of sparseness, using early termination increases the growth of cost associated with increasing the tuple size.

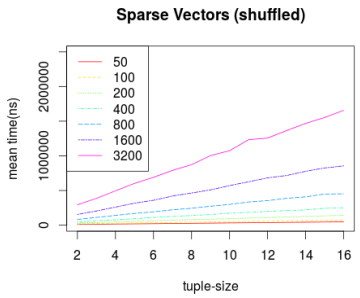
Figure 16 compares the growth associated with increasing the tuple size of each mechanism on unshuffled data. In this case we see that early termination has a positive effect. Those with early termination grow more quickly.



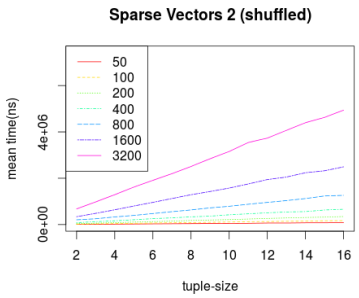
(a) Inverted Index



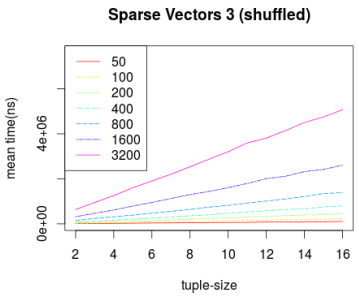
(b) Inverted Index ET



(c) Sparse Vectors

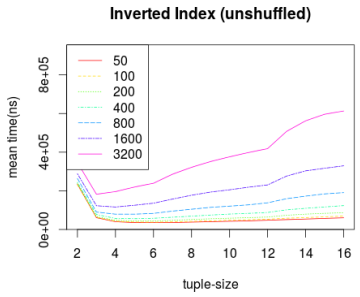


(d) Sparse Vectors 2

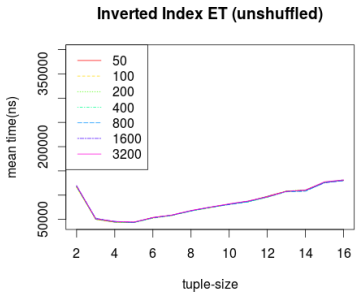


(e) Sparse Vectors 3

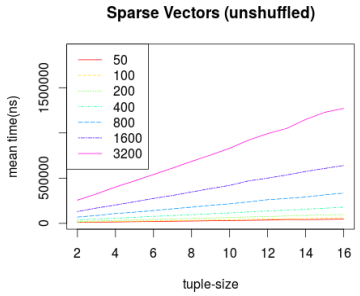
Figure 13: Effect of tuple size on the shuffled datasets for each of the evaluation techniques



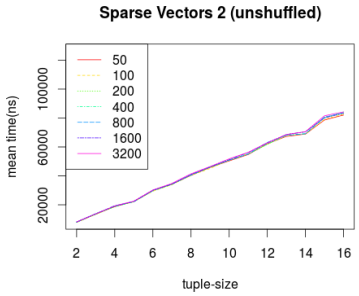
(a) Inverted Index



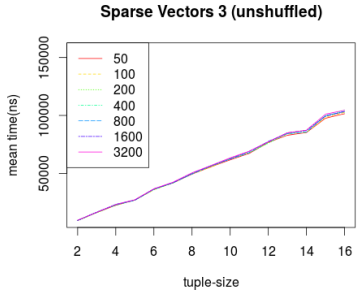
(b) Inverted Index ET



(c) Sparse Vectors

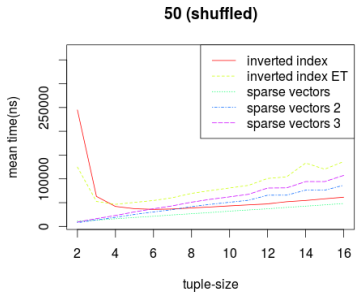


(d) Sparse Vectors 2

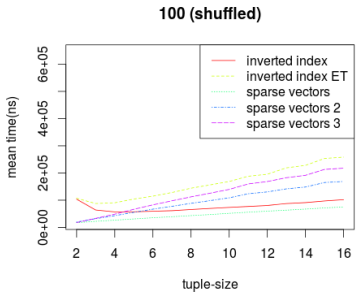


(e) Sparse Vectors 3

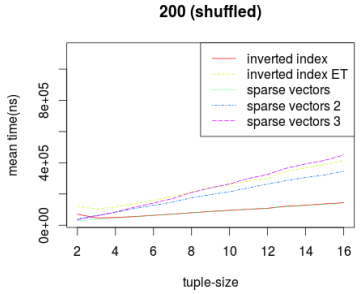
Figure 14: Effect of tuple size on the unshuffled datasets for each of the evaluation techniques



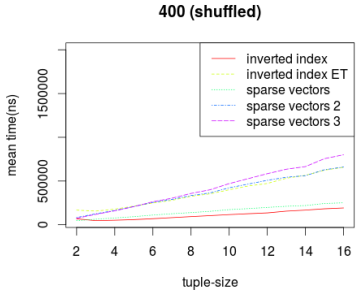
(a) 50 dims



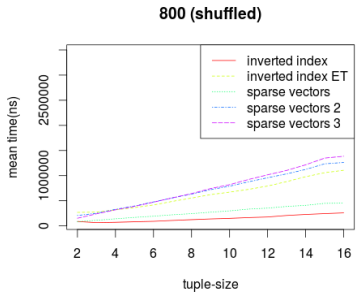
(b) 100 dims



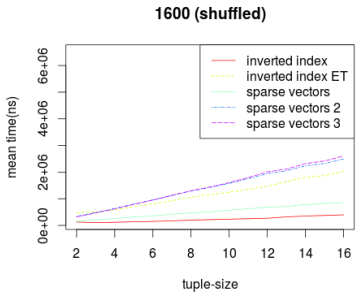
(c) 200 dims



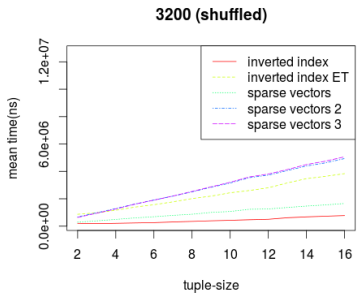
(d) 400 dims



(e) 800 dims

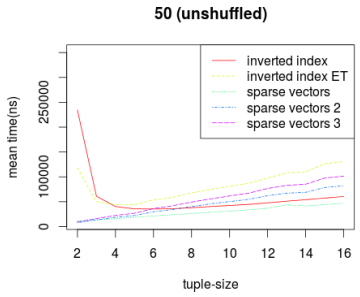


(f) 1600 dims

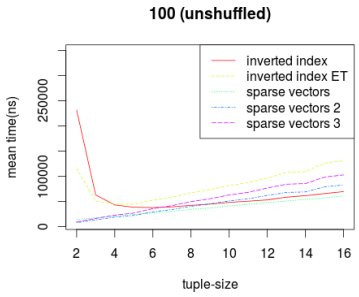


(g) 3200 dims

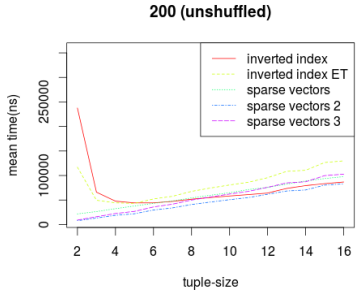
Figure 15: Shuffled



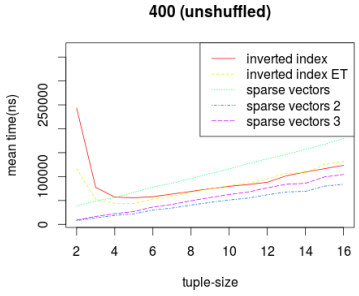
(a) 50 dims



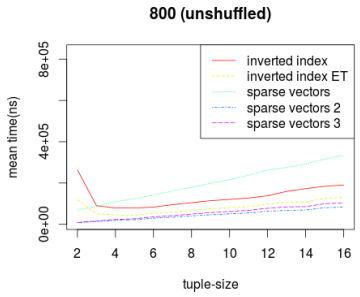
(b) 100 dims



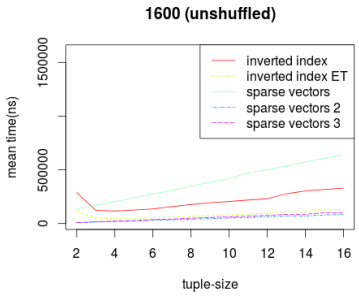
(c) 200 dims



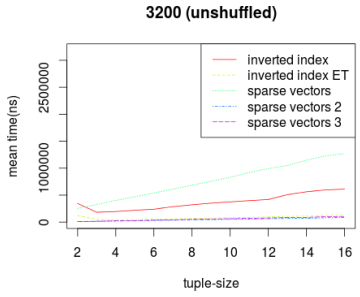
(d) 400 dims



(e) 800 dims



(f) 1600 dims



(g) 3200 dims

Figure 16: Unshuffled

5.4 CONCLUSION

The benefit associated with early termination and skipping dimensions clearly depends on the dataset being considered. We examined the two extremes: when all the data have the same dimensions occupied and when they are populated randomly. In the first instance, early termination clearly helped to reduce the cost of the calculation, but in the second the cost of maintaining the threshold calculation outweighed any saving. In reality, most datasets will lie somewhere between these extremes, and may require some analysis to determine whether it would be appropriate to use these mechanisms.

Part III

APPLICATIONS OF SED

Having explored the various properties of structural entropic distance we now shift our attention to the more practical aspect of applying it to common problems involving distance.

6

SIMILARITY SEARCH

6.1 SIMILARITY SEARCH

Collections of unstructured or semi-structured data, where semantics are implicit in the data itself differ from relational databases, where the semantics of the data is contained in their schema. Querying this type of data requires that *features* representative of the semantics must first be extracted before any kind of comparison can be made. Since it is often impossible to know *a priori* what features will be present (as this will depend, to a large extent, on the domain), a natural question is how similar the respective feature sets are. This is the essence of similarity search, which seeks to find other data which have similar semantics to a given data point.

Similarity search is increasingly playing a greater role in modern storage and retrieval techniques. In this paradigm, the similarity of a query object to the objects stored in the database form the basis of the search. The result of a query then is the set of all objects that are similar to the query object. More formally:

Definition 14. *Given a collection* X*, and a query element* q*, retrieve all elements* $x \in X$ *that are similar*[1] *to* q*.*

In chapter 2 we refine this notion of similarity using a mathematical notion of distance. Objects in a database can be thought of as points in a space that have a distance between them – where similar objects are near each other. The results of queries may be in terms of a region of the space, or as clusters within a region.

Applications for similarity search include: Optical character recognition; Statistical classification; Computer vision; Content-based image retrieval; Maximum likelihood decoding; Data compression; Recommendation systems; Contextual advertising and

1 We are deliberately vague about the definition of similar here, as there are numerous types of query available

behavioural targeting; DNA sequencing; Spelling checking; Plagiarism detection; and Cluster analysis. We now look more closely at some of these examples:

Example 1. *Multimedia document retrieval*

One of the most common applications of Similarity Search is the retrieval of multimedia documents. E.g face recognition, object recognition, fingerprint matching, voice recognition, and multimedia databases in general. In this case the raw content of the document is not used, rather documents are processed for features. Search is performed by comparing features. Consider images, features extracted may be colours, textures, or shapes, represented as vectors. One technique is to use each dimension of the vector to represent a colour and the size of each dimension is the number of pixels with that colour – effectively a histogram. The search then focuses on comparing the histograms.

Example 2. *Recommender Systems*

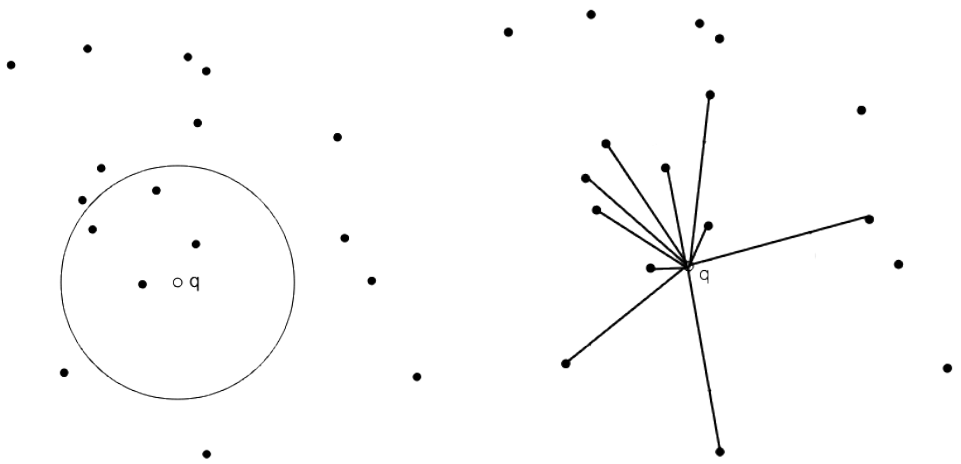
By maintaining a profile of their users, recommender systems are able to suggest something new to a user (eg., to buy a book or visit a web-site). In such cases the user's preferences are compared to other users' for similarity and suggestions are made on the basis that the user will like items that similar users do.

6.2 SEARCH QUERIES

In general, there are two main types of query that arise from using distance as a mechanism for similarity: A range query, or a nearest-neighbour search. A range query uses the distance function to effectively capture all elements that fall with in a particular radius of the query object.

Definition 15. *A range query is a function* $R : S \times \mathbb{R} \to S$ *such that* $R(q, r) = \{y : d(q, x) < r\}$ *where* $r > 0$

Definition 16. *A nearest neighbour query is a function* $NN : S \times \mathbb{Z}^+ \to S$ *where* $NN(q, k) = R \subseteq S$ *such that* $\forall x \in R, y \in S - R, d(q, x) < d(q, y)$ *and* $|R| = k$.



(a) Range query (b) Nearest-Neighbour query

Figure 17: Range query and nearest-neighbours query in two dimensional Euclidean Space.

6.3 THE COST OF SIMILARITY SEARCH

Having established the link between JSD and SED, we should now like to compare them. Since both give the same ordering, we must examine efficiency. We will compare them using intrinsic dimensionality (IDIM) and a probabilistic analysis designed to measure the cost and general notion of the tractability of similarity search techniques over high-dimensional metric spaces.

Intrinsic dimensionality has been introduced as a tractability measure[], and a lower bound for index cost has been established. In practice, however, we have found different approximately-Gaussian distance distributions with the same IDIM may have very different probability densities close to the origin, and this has a far greater implication on performance than the IDIM. IDIM is really meaningful only within the context of a close-to-perfect Gaussian distribution, to which many metric/data combinations do not conform.

To address these limitations, we have constructed a methodology to perform a probabilistic analysis over metric spaces. Like IDIM, it is based only on the distances over data, rather than the data itself. According to probability densities at both the median, and close to the origin. We show how this can be used to estimate the cost of a search operation over a given size of data. It can be calculated via random sampling

over a data set, and we have assessed performance predictions against measured performances on various data sets and found to be useful.

Many metric spaces over which similarity search is desirable suffer from the so-called curse of dimensionality. In the domain of Similarity Search, this refers to the effect whereby, as the number of dimensions increases, the distance between any two arbitrarily selected points is increasingly less likely to be relatively either small or large, and ever more distances become close to the median. As most measured distances become closer, any indexing mechanism becomes less effective.

As similarity search matures, ever more data collections and distance metrics become subject to these techniques; the question of whether a given combination of data and metric is likely to lead to tractable searching is therefore an important one.

Much research into the cost of search-based query is in the domain of complexity analysis, and does not take dimensionality into account. Our approach is more akin to abstract interpretation, with the intent of giving a realistic measurement of efficacy which can take into account issues such as the data size, the semantic effectiveness of the metric, and the query threshold chosen as an implication of these. Although independent of any particular indexing mechanism, the resulting measurement does seem to give a good indication of realistic query cost as measured against a number of different index structures.

In outline, our method for performing an estimate of effectiveness comprises:

- First, a probability density function $f(x)$ for the metric space $M = (X, d)$ is established by the repeated application of d over object pairs randomly sampled from X;
- Using f, the magnitude of the collection $|X|$, and the semantic accuracy of the metric d, a likely query threshold t is determined;
- Using f and t, the probability p of a successful pivot operation around the median distance is established;
- Using $|X|$ and p, a pragmatic upper bound of the likely number of distance calculations required at query time is established.

We will then show with a number of different datasets which distance metrics have the lowest upper bound cost for searching using a typical metric index.

6.3.1 *Indexing, pivoting and probabilities*

The majority of metric space indexing techniques rely upon the notion of pivoting, which in turn relies upon the triangle inequality property of the metric in question. To understand how well a given metric and data collection will perform, it is necessary to understand the probability of an indexing mechanism performing a successful prune of the search space at each step.

With respect to a pivot object p, a query object q, a query radius r, a pivot covering radius μ:

- if $d(p, q) - r > \mu$, then those points known to be within μ of p cannot be within r of q
- if $d(p, q) + r \leqslant \mu$, then those points known to be outside μ of p cannot be within r of q

The efficacy of any index structure built using these principles depend upon the probability of each of these two events. These probabilities are defined here as $p(e)$, the exclusion probability, and $p(i)$, the inclusion probability, respectively. From the structure of their definitions it is clear that both events are mutually exclusive, and so the probability of either one occurring can be expressed as $p(e) + p(i)$. Further, this sum is $\leqslant 1$, with the sum being 1 if and only if $r = 0$; thus, if the search is for the query object itself, or those within the same equivalence class, then the pivoting mechanism turns into a deterministic fetch. More commonly r will not be 0, in which case it is clear that smaller it is, the closer $p(e) + p(i)$ will be to 1, giving better query performance.

6.3.2 *Probability density functions and pivoting*

Probability density functions are mathematically quite different from histograms; a very good approximation to a pdf, however, can be obtained experimentally by constructing a fine-granularity histogram over a large number of sample measurements, dividing each count by the total number of measurements over the granularity of the histogram, and using interpolation to treat the outcome as a continuous function.

With respect to a probability density function $f(x)$ of a given distance metric over a data collection, the exclusion probability $p(e)$ can be quantified as:

$$p(e) = \int_{\mu+r}^{1} f(x) \tag{66}$$

and similarly the inclusion probability:

$$p(i) = \int_{0}^{\mu-r} f(x) \tag{67}$$

As $\int_0^1 f(x) = 1$ and the exclusion and inclusion events are mutually exclusive, the probability of either occurring is

$$p(e) + p(i) = 1 - \int_{\mu-r}^{\mu+r} f(x) \tag{68}$$

If the covering radius μ is chosen to be the median, this probability represents the likelihood of excluding half of the objects covered by that pivot.

For a distance function and data collection whose pdf approximates to Gaussian, this probability depends only on the standard deviation and query threshold; it is worth noting that it is independent of the actual value of the median, and therefore to a degree independent of the IDIM. It is very likely that the IDIM will be strongly related to the requirement for a given query threshold, a greater IDIM usually implying a greater threshold. It is, however, clear that the query threshold is the predominant factor with respect to the performance of any mechanism relying upon these probabilities.

6.3.3 *Query thresholds and PDFs*

While a Gaussian distribution is often a very good overall approximation for the probability of distance for a pair of random points drawn from a large space, it is never a good approximation for small distances. The reason is that Gaussian distributions are defined in the range $-\infty$ to ∞, but the outcome of a distance metric always has a probability of zero for values less than zero.

This distinction is important; even metrics that show very close correlation with Gaussian distributions across most of the range show markedly different behaviours close to the origin. A threshold distance for a large data collection for example may well be set so that one-millionth of the data is likely to be returned; that threshold may vary widely with different metrics that have the same median and standard deviation, and the Gaussian model is likely to place it to the left of the origin.

For analysis, we consider a balanced Vantage Point Tree structure constructed around the median distance of the metric space. A Vantage Point tree over the metric space $M = (X, d)$ is a binary tree where each node holds a single element of X. For every node holding a point x_i , every node (if any) within the left-hand subtree contains a point x_j such that $d(x_i, x_j) < \mu$ (where μ is the median of the range of d over X), and every node within the right-hand subtree contains a point x_j such that $d(x_i, x_j) > \mu$.

A VP-tree can be created by the following algorithm:

- A point added to an empty tree results in a single node containing that point.
- A point x_j added to an existing node containing point x_i at its head results in a new tree with that point added to the left branch if $d(x_i, x_j) < \mu$, and with that point added to the right branch otherwise.

A VP-tree for collection X is then constructed by taking all the points of X in an arbitrary order and adding them to an empty tree. To allow analysis, we make the following assumptions:

- The tree is perfectly balanced, i.e. the size of X is $2n - 1$, for some n, and every branch from root to leaf is the same length.

- Every node of the tree is constructed with respect to the same pivot value
- No further mechanism or strategy exists to optimise the tree or its indexing.
- For any three arbitrarily selected objects x_i, x_j and x_k, then $d(x_i, x_j), d(x_j, x_k)$ and $d(x_i, x_k)$ are all probabilistically independent of each other.

Given the probabilities $p(e)$ and $p(i)$ defined above, and a perfectly-behaved Vantage Point Tree, it is possible to derive a cost estimate for querying data collections of different sizes.

6.3.4 *Number of distance calculations*

The method of construction of the tree in conjunction with the triangle inequality property of allows the search space to be pruned in the manner given. To consider the quantification of this process, we consider the complements of the exclusion and inclusion probabilities defined above, $q(e)$ and $q(i)$, representing respectively the probability that the left and right subtrees will be visited by the search algorithm.

Now consider the probability of visiting each node in the tree. The probability of visiting the root node is 1, that of the left hand subtree is $q(e)$ and that of the right hand subtree is $q(i)$. More generally, for any node n, if the probability of visiting node N is $p(n)$, then the probability of visiting its left subtree is $p(n)q(e)$ and that of visiting its right subtree is $p(n)q(i)$. For the four nodes at level 2 of the tree, considered from left to right, the probability of visiting each is therefore: $q(e)^2$, $q(e)q(i)$, $q(i)q(e)$ and $q(i)^2$ respectively. Over many uses of this tree to perform a search, it is therefore the case that the mean number of nodes visited at this level will be

$$q(e)^2 + 2q(e)q(i) + q(i)^2 \tag{69}$$

At the third level of the tree this mean figure is

$$q(e)^3 + 3q(e)^2q(i) + 3q(e)q(i)^2 + q(i)^3 \tag{70}$$

It can be seen that these sums respectively represent $(q(e) + q(i))^2$ and $(q(e) + q(i))^3$.

By induction over the tree structure, the mean number of nodes to be visited at depth d is $(q(e) + q(i))^d$, and for a perfectly balanced tree of depth d the mean number of nodes to be visited for a whole traversal is therefore

$$\sum_{i=0}^{d} (q(e) + q(i))^i \tag{71}$$

which can now be written succinctly as a function over a metric space $M = (X, d)$: Let $f(x)$ be the probability density function of d over X, with median μ; let h be the height of a perfectly balanced VP tree containing X. Then for a query threshold r, let the query cost estimator C be defined as

$$C = \sum_{i=0}^{h} (q(e) + q(i))^i \tag{72}$$

$$= \frac{(q(e) + q(i))^{h+1} - 1}{q(e) + q(i) - 1} \tag{73}$$

$$= \frac{\left(\int_0^{\mu+r} f(x) + \int_{\mu-r}^{1} f(x)\right)^{h+1} - 1}{\left(\int_0^{\mu+r} f(x) + \int_{\mu-r}^{1} f(x)\right) - 1} \tag{74}$$

$$= \frac{\left(1 + \int_{\mu-r}^{\mu+r} f(x)\right)^{h+1} - 1}{\int_{\mu-r}^{\mu+r} f(x)} \tag{75}$$

The importance of the effect of r on the calculation is clearly highlighted, given that h is likely to be in the region of 20 - 30 for many applications of similarity search.

For a space with n nodes the maximum height h of a binary tree is $\log_2(n + 1) - 1$.

6.3.5 *Verification of the cost function*

Experiment with Vantage point tree to show that this is a reasonable cost function.

6.4 COMPARISON OF DISTANCE METRICS

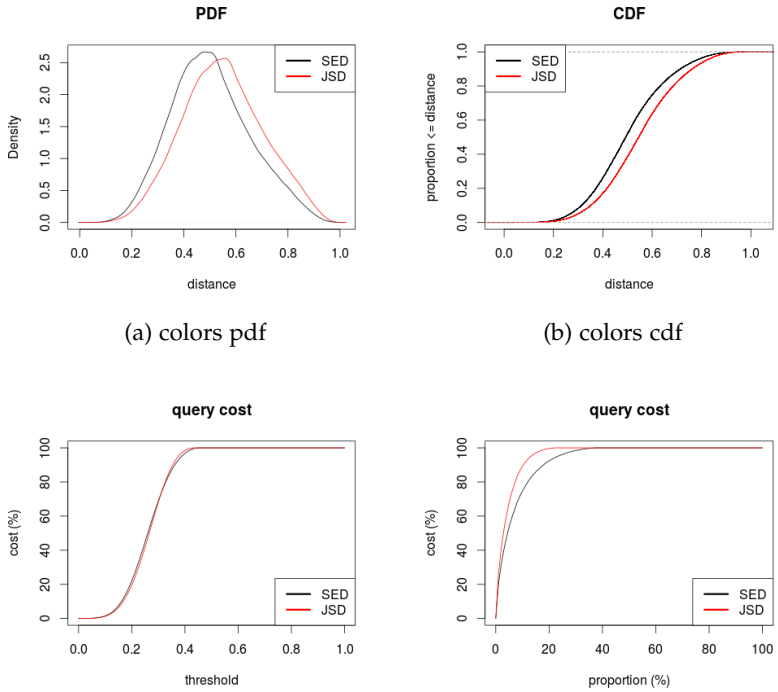
A central requirement of metric space indexing structures is that the distance metric has the triangle inequality property. In their definitional form none of NCD, CPD, SED nor Jensen-Shannon divergence obeys triangle inequality, and are thus not proper distance metrics. Raising SED to the power of 0.48 achieves triangle inequality for vectors not generated in this manner. Recently, [] have shown that the square root of Jensen-Shannon divergence is also a proper metric. We know of no transformation to NCD or CPD to allow their use in a metric space indexing structure and thus exclude them from the following comparison.

We applied the cost function described above on both generated and real-world data to compare the techniques described in this chapter.

	IDIM		1%	2%	3%	4%	5%
$SED^{0.48}$	8.49854	threshold	0.1517	0.1613	0.1683	0.1737	0.1800
		cost	33504	39886	45346	49983	55714
$JS^{0.5}$	8.50651	threshold	0.1676	0.1786	0.1866	0.1927	0.1999
		cost	32851	39060	44441	49026	54709

Table 2: Generated Tree data, depth 4, branching-factor 10

Not all datasets are as uniform as this, however. The next dataset, also generated, allowed much greater variation among the data points. Figure **??** gives a more holistic view of these data. Looking at the entire distribution at once, first, the probability density function allows the calculation of the IDIM, and allows us to gauge the dimensionality visually; secondly, the cumulative density function allows us to see what proportion of the data fall below a certain distance, giving us the opportunity to pick a threshold distance for a range query; thirdly, we can calculate the cost function in terms of query radius; then finally, we can perform a reverse lookup on the CDF to find what query radius will return what proportion of the data, then apply this to the cost function to view it in terms of expected return set size. This allows a fair comparison between metrics, since the absolute values of the distances cannot be used to perform a comparison.



(a) colors pdf

(b) colors cdf

(c) colors threshold cost

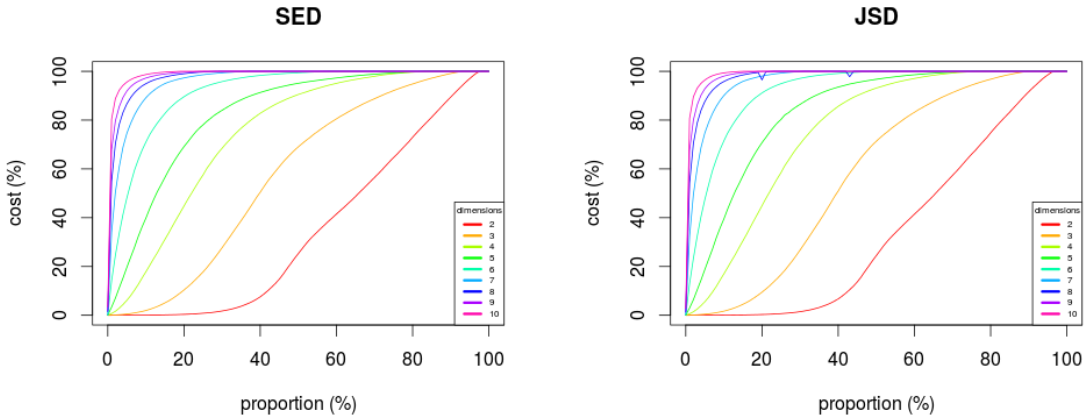
(d) colors proportion cost

Figure 18: Colors dataset, showing the predictive costs of searching a metric index from a sample of distance. Top left is the probability density function, top right is the cumulative density function, bottom left is estimated cost of searching with a particular query radius, and bottom right is the estimated cost of performing a query that is expected to return a given proportion of the data in the index

Our final dataset was a set of 112,544 color histograms (112-dimensional vectors) from an image database available from SISAP[2]. Figure **??** displays the same process as the last dataset. This time SED was required to have triangle inequality enforced.

In this real-world dataset, there was a clear benefit to using SED over JSD; the cost function grew at a substantially slower rate. And in general, SED is a better choice for metric search.

2 http://www.sisap.org



(a) SED

(b) JSD

Figure 19: Metric Search Cost – Generated vectors

7

INFORMATION RETRIEVAL

Information retrieval is primarily concerned with producing a descending order of ranked documents in response to an information need expressed as a short keyword based query[**?**]. Models are produced which define a scoring function which should be based on a formalisation of relevance. Good empirical performance, however, comes from heuristic modifications to the model. Indeed, the most successful model all utilise a combination of common heuristics, e.g., Inverse Document Frequency (IDF), document length normalisation, parameters for each collection.

In this chapter, we describe an un-parameterised ranking model based purely on term frequencies, which uses the concept of probabilistic document generators. We produce a function based on SED to estimate the probability that a given document was produced by one of two generators: A *Subject-free* generator, based on the frequency of terms within a corpus, and a *subject-based* generator, where a subject is defined by a set of key terms. Documents relevant to a subject are measurably more likely to have been generated by the generator for that subject than by the *subject-free* generator, so we rank documents according to the difference between these probability estimates.

7.1 RANKING MODELS

7.1.1 *Vector space model*

The cosine of the angle between vectors is often used as a similarity function:

$$\mathrm{Sim}(\mathbf{d}, \mathbf{q}) = \frac{\langle \mathbf{d}, \mathbf{q} \rangle}{\|\mathbf{d}\|_2 \|\mathbf{q}\|_2} \tag{76}$$

$$= \frac{\sum_{t \in d \cup q} tf_{t,d} \cdot tf_{t,q}}{\sqrt{\sum_{t \in d} tf_{t,d}^2} \cdot \sqrt{\sum_{t \in q} tf_{t,q}^2}} \tag{77}$$

where $tf_{t,d}$ and $tf_{t,q}$ are the count of occurrences of t in document d and query q, known as term frequencies. Term weights are usually adjusted by multiplying each term frequency by the Inverse Document Frequency (IDF) of the term, where IDF is calculated as

$$\mathrm{IDF}(t) = -\log(df_t / N) \tag{78}$$

N is the number of documents in the corpus and df_t is the number of documents containing term t. IDF(t) is therefore the information gained by observing a document containing term t; if the term appears in a most documents there is very little information gained from observing it.

7.1.2 *Probabilistic model*

Under the probalistic model, the probability of a document being relevant given the need expressed by the query is the value of interest[**?**]. Binary Independence Model (BIM), achieves this by ranking documents by $P(Relevant|\mathbf{d}, \mathbf{q})$; through algebraic manipulation, the same ordering is produced by the easier calculation,

$$R(\mathbf{d}, \mathbf{q}) = \sum_{t \in d \cap q} \log \frac{p_t}{1 - p_t} - \log \frac{u_t}{1 - u_t} \tag{79}$$

where p_t is the probability of a term t appearing in a document relevant to the query, and u_t the probability of t appearing in a non-relevant document; that is, if s is the number of relevant documents containing term t, and S is the total number of relevant documents, then

$$p_t = \frac{s}{S} \quad \text{and} \quad u_t = \frac{df_t - s}{N - S} \tag{80}$$

In information theoretic terms:

$$\sum_t I(1 - p_t) + I(u_t) + I(1 - u_t) - I(p_t) \tag{81}$$

The total information gained from observing a term t is the information gained when d is relevant but does not contain t, plus the information gained when d non-relevant but contains t, plus the information gained when d is non-relevant and does not contain t, minus the information gained when d is relevant and contains t.

In practice, these probabilities are problematic to calculate: under the assumption that the number of relevant documents is a very small proportion of the corpus, u_t is estimated as $\frac{df_t}{N}$; whereas, empirically, $\frac{1}{3} + \frac{2}{3} \cdot \frac{df_t}{N}$ has been found to be a plausible estimate of p_t.

The BM25 model [**?**] is essentially the sum of TF $\times$ IDF values, with the term frequency weighted as follows:

$$BM25 = \sum_{t \in q} IDF(t) \cdot \frac{(k_1 + 1)tf_d}{k_1 \alpha(b) + tf_d} \cdot \frac{(k_3 + 1)tf_q}{k_3 + tf_q} \tag{82}$$

where k_1 and k_3 are positive tuning parameters that calibrate the document and query term frequency scaling; the function

$$\alpha(b) = 1 - b(1 + \frac{L_d}{L_{ave}}) \tag{83}$$

corresponds to document length normalisation, where $0 \leqslant b \leqslant 1$; L_d is the length of d, and L_{ave} is the average document length in the corpus. Full length normalisation is applied when $b = 1$ and none is applied when $b = 0$.

7.1.3 *Language model*

Query Likelihood Language Models (LM) have a much more principled basis and are able to perform well compared to TFIDF and BM25. These assume that the query is drawn randomly according to the LM of a document and therefore seek to determine the likelihood that the query was generated given the LM of each document, i.e.

$$Score(q|d) = P(q|\theta_d) \tag{84}$$

where θ_d is the LM of document d [**? ? ?**].

The LM for each document is usually calculated as the maximum likelihood (ML) estimate of the probability $P(t|d)$ of a vocabulary term t given the document d, which is simply the count of term t in the document divided by the length of the d. In order for these models to function correctly (to avoid o probabilities), however, it is necessary to introduce smoothing to the ML estimates [**?**] thereby giving non-zero probability to unseen words. Various smoothing methods have been proposed, the most successful of these assumes a Dirichlet prior distribution over the document LM multinomials, and is therefore referred to as Dirichlet smoothing:

$$P(q|d) = \prod_{t \in q} \frac{tf_{t,d} + \alpha \frac{cf_t}{T}}{L_d + \alpha} \tag{85}$$

7.2 HYBRID VECTOR-LANGUAGE MODEL

From document d we construct a language model θ_d where $P(t|\theta_d)$ denotes the probability of term t occurring in document d, given by the maximum likelihood estimate $P(t|\theta_d) = \frac{tf_{t,d}}{L_d}$. We build a language model θ_c for the entire corpus too in a similar fashion; the probability of t occurring in the corpus $P(t|\theta_c) = \frac{cf_t}{T}$.

A notional document g of length n is constructed by making n successive calls to a term generator $\mathcal{G}$, driven by the language model θ_g, where $\mathcal{G}$ randomly returns a term t with probability $P(t|\theta_g)$.

7.2.1 *Probability Estimation*

Consider a similarity function over term vectors; this may be used to compare pairs of actual or generated documents, and also as an estimate of the probability $P(d|\theta_g)$ that a generator g was used to produce document d. This will allow us to meaningfully compare a document against different generators in order to find the one most likely to have generated it.

Consider now a notional set of documents N, which are not relevant to *any* particular subject, we will refer to these as *null* documents. The terms of such documents are modelled by the language model θ_c.

7.2.2 *Documents, Topics and Key Terms*

In documents relevant to one or more particular subject, each subject will be related to a set of key terms within the corpus; the relative frequency of terms appearing within each such document is greater than that of the same term within the corpus.

Such documents are thus characterisable by key terms $K = \{k_1, \ldots, k_n\}$, where each k_i is a term in the corpus that acts as a keyword for the subject. The relative frequency of these terms increases over and above its relative frequency within the corpus: for example, a document about *red aardvark*s may well have extra instances of that phrase, but a significantly longer document is likely to have the terms in other contexts as well.

All other terms in the document have a decreased relative frequency, although this would be a much less obvious effect especially in long documents; this is, in some sense, related to van Rijsbergen's cluster hypothesis [**?**]. We assume terms to be distributed differently in relevant and non-relevant documents as a consequence of the existence of topics. We do not, however, assume that documents that are similar to each other have high likelihood of being relevant to the same queries.

7.2.3 *Relative Probability Ranking*

For a subject described by a set of key terms K, any document d about a subject characterised by K will likely be more similar to a generated document g than to a null document n:

$$\mathrm{Sim}(d, g) > \mathrm{Sim}(d, n) \tag{86}$$

whereas, if document d is about a different subject, with no overlapping key terms, then it is more likely to be more similar to n than g. As a corollary, consider the function

$$R(d, g) = \mathrm{Sim}(d, g) - \mathrm{Sim}(d, n) \tag{87}$$

to be a ranking function. For all terms in each document that are *not* stated as key terms in a search, we assume that the term frequency is in ratio with the corpus term frequency. This self-same assumption is made in other probabilistic models such as the BIM model [**?**]. Thus, we consider notional document vectors that maintain their term frequencies for terms in the query, but otherwise effectively lose all other information. Our comparison to find the more relevant document has now become the higher value of $R(d_1, g)$ and $R(d_2, g)$.

We use the statistical distances as the basis of this ranking function as follows:

$$\mathrm{SED}(d, g) = 2^{\mathrm{JSD}(d,g)} - 1 \tag{88}$$

where, if we recall from chapter **??**, JSD can be efficiently computed as:

$$\mathrm{JSD}(d, g) = 1 - \frac{1}{2} \sum_{t \in \mathbb{C}} \mathcal{F}(P(t|\theta_d), P(t|\theta_g)) \tag{89}$$

and

$$\mathcal{F}(x, y) = (x + y) \log_2(x + y) - x \log_2 x - y \log_2 y \tag{90}$$

Since they have no effect on the ordering produced by the ranking function, we remove all constants, giving

$$D(d, g) = -\sum_{t \in C} \mathcal{F}(P(t|\theta_d), P(t|\theta_g)) \tag{91}$$

and $1 - D(d, g)$ as an estimate of the probability that d was generated by G. Thus we use the similarity function,

$$Sim(d, g) = \sum_{t \in C} \mathcal{F}(P(t|\theta_d), P(t|\theta_g)) \tag{92}$$

and the ranking function becomes

$$R(d, g) = \sum_{t \in K} \mathcal{F}(P(t|\theta_d), P(t|\theta_g)) - \mathcal{F}(P(t|\theta_d), P(t|\theta_c))) \tag{93}$$

For each term in the query, we require only the term frequency in the corpus, from which the term frequency in the notional subject generator is calculated, and the term frequency in the document. As no global parameter such as document length is required per document, this equation is actually slightly more space-efficient than many other methods, although the run-time calculation is a little more expensive.

If an absolute value is required, to allow comparisons of different queries, then the residual terms can also be calculated without greatly increasing the run-time cost.

7.3 RETRIEVAL PERFORMANCE

7.3.1 *Collections and Models*

To evaluate our model against standard IR baselines we used 3 test collections from the Text REtrieval Conference (TREC), the first of which was the Text Research Collection Volumes 4 & 5, which includes material from the Financial Times Limited (1991, 1992, 1993, 1994), the Congressional Record of the 103rd Congress (1993), the Federal Register (1994), the Foreign Broadcast Information Service (1996) and the Los

Angeles Times (1989, 1990). The second was the WT10G collection, which is a general Web crawl and the third was the Blogs06 collection, a large collection of over 100,000 blogs. For the first we used topics 301-400 from TREC-6 and TREC-7, for the second we used topics 501-550 from the TREC Web track and for the third topics 851-900 from the blog track. These collections were chosen as they are standard in IR and are quite different from one another. Having results from all 3 collections gives us insights into how generalisable a retrieval function might be.

Each topic provides three fields. A title, a description, and a narrative; we used short queries (where only the title was used) and long queries (where all three fields were used).

All of our experiments were carried out using the Terrier IR platform [**?**], which provides out-of-the-box implementations of our comparison metrics: TF/IDF with Robertson's TF (TFIDF), BM25, and LM with Bayesian smoothing and Dirichlet prior (DirichLM). We used default parameters for each of the models found in the literature, which are generally quite well optimised for TREC experimentation; BM25 $k_1 = 1.2$, $k_2 = 8$, $b = 0.75$; Robertson TF/IDF $k_1 = 1.2$, $b = 0.75$; Dirichlet LM $\mu = 2500$. Although these values may not give strictly optimal retrieval performance they provide a fairer comparison with our parameterless function. We refer to the proposed model as Unified Probability Model (UPM) throughout.

The Terrier platform returns a ranked list of the top 1000 documents for each query with its associated score. Using supplied relevance judgements we were able to determine which of these documents were relevant.

7.3.2 *Results*

Figure 20: Precision/recall, short queries, Trec6-7

Figure 21: Precision/recall, long queries, Trec6-7

Table 3 gives an overview of the performance of all of the models over the 3 collections for both short and long queries. We use 3 standard metrics in IR to determine the

	Trec 6-7			Trec 6-7 Long		
Metric	mAP	mRR	nDCG	mAP	mRR	nDCG
BM25	0.292	0.603	0.621	0.237	0.677	0.587
DirichLM	0.275	0.543	0.597	0.196	0.536	0.543
UPM	0.280	0.575	0.603	0.219	0.626	0.560
TFIDF	0.294	0.604	0.623	0.232	0.672	0.582
	WT10G			WT10G Long		
BM25	0.250	0.584	0.596	0.184	0.553	0.522
DirichLM	0.249	0.632	0.597	0.144	0.460	0.474
UPM	0.189	0.468	0.539	0.156	0.486	0.480
TFIDF	0.248	0.578	0.594	0.178	0.535	0.512
	BLOG06			BLOG06 Long		
BM25	0.429	0.695	0.793	0.362	0.770	0.747
DirichLM	0.457	0.720	0.802	0.302	0.729	0.711
UPM	0.318	0.413	0.713	0.306	0.687	0.704
TFIDF	0.430	0.695	0.793	0.356	0.768	0.740

Table 3: Performance of models over TREC 6 and 7, WT10G and Blogs06 collections. First, using short queries and secondly, using long queries

Figure 22: Precision/recall, long queries, WT-10G

relative performance of the competing models: mean average precision (mAP), mean reciprocal rank (mRR), and mean normalised discounted cumulative gain (nDCG).

Considering the short queries, the UPM model performs well on Trec6-7, clearly outperforming DirichLM and approaching the performance of the other models. Performance is worse on the WT10G collection, where the 3 other models return similar performance figures, and especially poor on the Blogs06 collection. However, when looking at the long queries we find that UPM is able to significantly outperform DirichLM for the first 2 collections and achieves similar performance for the Blogs06 collection. These results suggest that UPM is able to perform best in situations where both the documents in the collection and the queries are quite long.

The new model performs poorly on the Blogs06 collection where the documents are generally much shorter and more variant. This is likely because the other models

Figure 23: ROC Curve, long queries, Trec6-7

all include some form of document length normalisation, preventing shorter documents from being penalised. In addition to this the smoothing of terms, which is explicit in the LM and implicit in the others, is particularly important in the case of short documents and less so for longer ones. In comparison with DirichLM, UPM returns particularly good mRR scores, indicating that it is able to more consistently rank relevant documents higher in the ranked list. Figures 20 and 21 depict this performance difference with UPM achieving much higher precision values over the early ranks.

The ROC curve for long Trec6-7 queries (Figure 23) shows very little discrimination between all four models, indicating that each is as capable as the other at correctly classifying relevant documents. The curves for the other test collections are similar and not included for the sake of brevity[1].

7.3.3 *Comparison with Base Models*

As mentioned, the ranking we have tested here is the pure form of the concept, before any heuristics or normalising factors have been applied. At time of writing these have not yet been investigated, but there is no reason to suppose that they do not exist. It is worth noting the huge improvement that has been made over the other pure forms of ranking functions through intensive investigation over, in some cases, decades.

To check the validity of the UPM base model, we compared it against the corresponding base models of the other ranking functions before heuristics are applied. For TF-IDF this is (essentially) cosine distance; for BM25 we used the term frequency multiplied by the IDF component derived from the Binary Independence Model, and for LM we summed the relative term frequencies.

Figure 24 shows the PR chart for these outcomes over the Trec6-7 short queries, which gives a taste of how much better the pure form of UPM is over the others. This is by no means the most favourable comparison; UPM still acts relatively better over

1 but will be available via URL

the long queries, and in fact is by far the best ranking over all the query sets with the exception of short queries over the BLOG06 collection, where TF-IDF and BM25 still outperform it.

Figure 24: Precision/recall base functions, short queries, Trec6-7

7.4 CONCLUSION

We have described a new retrieval model for ad hoc ranking tasks based on a probabilistic semantics and have shown that its performance is comparable to 3 highly competitive baselines. Most notably, the model was shown to outperform the frequently-used Language Model with Dirichlet smoothing in 4 out of 6 test collections. The performance of the new model is particularly strong for long queries and long documents. While the model is not able to conclusively out-perform all baselines, its performance is extremely encouraging, given that it has not been extensively adapted to the ad hoc ranking task and a number of standard IR heuristics have not yet been applied to it. This is an important point since the other ranking models have been refined over many years, particularly in order to yield good results for the TREC collection, resulting in their current form. The model is based on a simple intuitive idea and in its current form is completely unadulterated, unlike the other retrieval models which often diverge significantly from their initial theoretical base.

It is interesting to compare the underlying hypothesis of the UPM model with the LM model; the former is based on the probability of query terms occurring within different documents, while the latter is based directly on the probability of a query being relevant to a document. This would imply directly that UPM may be a better model over longer documents and queries, and that the LM may be better over shorter documents and queries. Our results strongly suggest that this is indeed the case and we would therefore expect that UPM would be an excellent model for tasks such as patent retrieval, where the queries are often entire documents.

8

CLUSTERING, CLASSIFICATION AND OUTLIER DETECTION

Cluster analysis uses a distance function to partition a vector space into groups of vectors called clusters that are mutually similar. The results of these clusters are frequently used for metric indexing. In this chapter, we focus on two categories of algorithms to achieve this task that have a strong resemblance to indexing mechanisms used for similarity search: agglomerative and partitional, looking at the clusterings produced with SED as the distance metric, and how the multi-way generalisation (MSED) can be used to produce clusters that are farther apart and whose elements are more mutually similar.

In the sections that follow, we devise an algorithm for hierarchical clustering based on MSED; an algorithm for selecting the seeds to the KMeans algorithm using MSED; and a cluster validity measure based on MSED. We show that hierarchical clusterings produced by our algorithm form more dense clusters that are farther apart than those produced by other hierarchical algorithms, using both existing cluster validity measures and our own. We also show that the seeds produced by our selection algorithm are more divergent than existing mechanisms employed by KMeans, although whether this actually helps the KMeans algorithm depends on the properties of the data. Producing tight clusterings that are clearly separated is useful for metric indexing since it allows greater pruning opportunity.

8.1 CLUSTERING TECHNIQUES

Two forms of clustering are of particular interest because of their likeness to metric indexes: agglomerative methods and partitional methods. In this section we introduce an agglomerative algorithm, a seed selection algorithm for partitional methods and an internal cluster validity measure all of which use MSED at their core.

Table 4: Merge criteria for different hierarchical algorithms

method	merge criterion
Single-Linkage	$\arg\min_{(A,B)}\{d(a,b) : a \in A, b \in B\}$
Average-Linkage	$\arg\min_{(A,B)} \left\{ \frac{1}{\lvert A\rvert\lvert B\rvert} \sum_{a\in A} \sum_{b\in B} d(a,b) \right\}$
Complete-Linkage	$\arg\min_{(A,B)} \{\max\{d(a,b) : a \in A, b \in B\}\}$
Centroid-linkage	$\arg\min_{(A,B)} \left\{ d(c_a, c_b) : c_a = \frac{\sum_{a\in A} a}{\lvert A\rvert}, c_b = \frac{\sum_{b\in B} b}{\lvert B\rvert} \right\}$
Ward's Method	$\arg\min_{(A,B)} \left\{ d^2(c_a, c_b) \frac{\lvert A\rvert\lvert B\rvert}{\lvert A\rvert+\lvert B\rvert} : c_a = \frac{\sum_{a\in A} a}{\lvert A\rvert}, c_b = \frac{\sum_{b\in B} b}{\lvert B\rvert} \right\}$

8.1.1 *Agglomerative Methods*

Agglomerative methods are bottom up, each point starts in its own cluster and pairs of clusters (A, B) are merged in a greedy fashion according to some merge criterion to produce a new cluster. The clustering ends when the required number of clusters exist or continues until all points are in the same cluster, at which point a tree can be produced showing a hierarchy of clusters. Table 4 shows the most commonly used merge criteria for hierarchical clustering.

MSED can be used to produce another clustering criterion in the same vein:

$$\arg\min_{(A,B)} \{\text{MSED}(A \cup B)\}$$

which joins the two clusters whose union has the smallest divergence. Intuitively, this makes sense, we want to keep the clusters as tight as possible. Given the fact that MSED correlates highly with mean intra-cluster distance, it would appear to be a faster version of Average-Linkage, but there is a subtle difference: Average-Linkage does not consider the mean distance in the union of candidate clusters, rather it considers the mean distance on the Cartesian product of the two candidate clusters.

8.1.2 *Partition Methods*

Partition methods divide up the space into partitions using hyperplanes. A set of seeds are selected initially and for each seed a cluster is defined consisting of all points that are closer to that seed than any other.

The KMeans algorithm is one such method. It begins in the usual way by selecting seeds, then partitioning the space by assigning each remaining point to its closest centroid. For the next iteration it selects as seeds the centroids of each cluster and repartitions the space. This process reiterates until there is no improvement according to some stopping criterion. Here we can make use of MSED: stopping when the sum of cluster divergences does not decrease.

KMeans, however, is highly sensitive to the initial set of seeds[**?**]. Although seeds are usually chosen at random, a selection algorithm, K-Means++, aims to provide points that are far apart with a high degree of probability as shown in algorithm 1.

Algorithm 1 KMeans++

$C \leftarrow$ a uniformly random point from X
while $|C| < k$ **do**
  $x \leftarrow$ a point p with probability $\propto \min_{c \in C} d^2(p, c)$
  $C \leftarrow C \cup \{x\}$
end while

Seed selection is another candidate application for MSED. Clustering is an expensive operation, and we do not want to cause unnecessary extra expense, so a greedy algorithm (algorithm 2) presents itself naturally as a fast solution that may be good enough: Iteratively, pick a point that maximizes the divergence of the set of

Algorithm 2 MSED seed selection

$C \leftarrow$ a uniformly random point from X
while $|C| < k$ **do**
  $C \leftarrow C \cup \{\arg\max_{x \in X \setminus C} \mathrm{MSED}(C \cup \{x\})\}$
end while

seeds. From inspection, this algorithm is at least as fast as KMeans++, asymptotically; KMeans++ has to calculate the squared distance of every point to each of the k seeds, while algorithm 2 has to calculate the MSED value of the set of seeds together with every point for each of the k seeds.

8.1.3 *Cluster validation*

Assessing validity of a clustering presents itself with many difficulties; the requirements, by necessity, are governed by the dataset being clustered. There are, however, two types of cluster validation technique in common usage: internal and external. Internal measures evaluate the clustering solely on the data that was clustered. External measures, rely on class labels produced by humans to evaluate the clustering.

Since we are examining clustering techniques in the context of metric indexing, external measures provide little guidance with respect to our notion of a good clustering.

8.1.3.1 *Internal cluster measures*

The following measures produce scores that favour high similarity within cluster and low similarity between clusters. One major drawback of internal measures is that they are biased towards algorithms that use same cluster model. Table 12 shows two such commonly used measures: The Davies-Bouldin Index, which examines all pairs of clusters and produces a score based on the ratio of the sum of average distance-to-centroid and the distance between cluster centroids; lower scores are indicative of better clusterings. The Dunn index, meanwhile, is based upon finding the minimum ratio of distance between clusters and the maximum of intra-cluster distances; here, higher scores are better.

One problem with both of these measures is that they are too slow to be of much use during clustering as, for example, a stopping criterion in partitional methods such as KMeans. We define another validation method using MSED with much lower computational complexity:

$$D = \frac{1}{n} \sum_{i=1}^{n} \mathrm{MSED}(C_i)$$

where C_i is cluster i. This is related to the stopping criterion that we suggested be used with KMeans, however we now use the average instead of the sum to allow

Table 5: Internal cluster validation metrics: $d(i,j)$ is the distance between clusters C_i and C_j, and $d'(C_k)$ is the intra-cluster distance for cluster C_k; c_i is the centroid for cluster C_i and σ_i is the average distance in cluster i to centroid c_i.

method	score
Dunn index	$\min_{1\leqslant i\leqslant n}\left\{\min_{1\leqslant j\leqslant n, i\neq j}\left\{\frac{d(i,j)}{\max_{1\leqslant k\leqslant n} d'(C_k)}\right\}\right\}$
Davies-Bouldin Index	$\frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left(\frac{\sigma_i+\sigma_j}{d(c_i,c_j)}\right)$

clusterings with different numbers of clusters to be compared. As one might expect, lower scores are better.

8.2 RESULTS

To evaluate the proposed techniques, experiments were performed on synthesised data and several real world datasets. First, we examine the cluster validation scores for the measures described in the previous section. Secondly, we evaluate the seed selection mechanism and determine its effect on the KMeans algorithm.

8.2.1 *Datasets*

8.2.1.1 *Synthesised data*

In designing an artificial dataset, we require that the data fall into separable clusters, rather than random points with no pattern. As such, 300 points were generated as follows: each dimension is a random gaussian in the range [number of dimensions, 100 + number of dimensions] except for one dimension, i, chosen uniform-randomly in [0, 100] to bring that point into a cluster C_i – the number of clusters was therefore the same as the number of dimensions. A number of datasets were generated with increasing dimensionality; the number of dimensions chosen respectively (and therefore cluster sizes) being 2, 4, 8, 16, 32, 64.

8.2.1.2 *Real world data*

It is important to consider that feature scales that are not consistent with their importance can distort results, however, SED requires normalized vectors, which is sufficient to mitigate this effect. In total, we made use of five real world datasets, all of which were obtained from [**?**].

IRIS Fisher [**?**] used this dataset to compare clustering algorithms. It contains 150 data points each with 4 features in 3 clusters, 50 points in each cluster. Each cluster represents one kind of Iris flower.

GLASS Created by German and donated to [**?**] by Spiehler. Motivated by criminological investigation since glass left at the scene of the crime can be used as evidence. It contains 214 data points each with 9 features in 6 classes of glass, distributed as follows: 70 float processed building windows, 76 non-float building windows, 17 float processed vehicle windows, 13 containers, 9 tableware, and 29 headlamps.

THYROID First used in [**?**]. Five laboratory tests were used to try to predict a patient's thyroid class as euthyroidism, hypothyroidism or hyperthyroidism. The diagnosis (the class label) was based on a complete medical record, including anamnesis, scan etc. It contains 215 samples each with 5 features in 3 classes, distributed as follows: 150 normal, 35 hyper, 30 hypo.

WINE These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. It was used in [**?**]. It contains 178 samples with 13 features in 3 classes, distributed as follows: 59 class 1, 71 class 2, 48 class 3.

WDBC Used by [**?**]. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. It contains 569 samples each with 30 features in 2 classes, distributed as follows: 357 benign, 212 malignant.

8.2.2 *Validation*

8.2.2.1 *Davies-Bouldin Index*

Remembering that with the Davies-Bouldin metric, lower scores are better, Table 6 shows that on the synthesised datasets, Divergence-linkage scores the lowest. Average-linkage, Single-linkage and Centroid-linkage score slightly higher, while KMeans, Ward's method and Complete-linkage score significantly higher.

For the real world datasets – Figure **??** shows the Davies-Bouldin Index score for each of the clustering mechanisms on each of these datasets – a similar picture emerges. Divergence-Linkage is lowest in all datasets except Iris where it is second lowest. Average-linkage performs similarly to divergence-linkage in all cases, but has slightly higher scores – this is no doubt a result of their similar definitions. Single-linkage scores well with most datasets, but on the wine dataset it has an unusually high score. KMeans and Complete-linkage score poorly with high scores on most datasets. Ward's method, too, has comparatively high scores, especially on the Glass dataset. Centroid-linkage scores are unremarkable and lie mostly in the middle.

8.2.2.2 *Dunn Index*

Table 6, again, shows Divergence-linkage and Single-linkage performing best with the Dunn Index on synthesised data. This time, however, Centroid-linkage also scores well. With this measurement the difference between Divergence-linkage and Average-linkage is highlighted, with the latter scoring significantly worse. The other clustering mechanisms, KMeans, Ward's method and Complete-linkage, in common with the Davies-Bouldin index are the least performant.

Table 6: Scores for the three internal measures on the synthesised datasets. Average and standard deviation of scores over all datasets

Algorithms	Davies-Bouldin	Dunn	Avg Divergence
SingleLinkage	0.620 ± 0.186	1.846 ± 0.934	0.00077 ± 0.00122
AverageLinkage	0.611 ± 0.225	1.383 ± 0.937	$\mathbf{0.00026 \pm 0.00056}$
CompleteLinkage	1.902 ± 0.775	0.993 ± 0.899	0.00835 ± 0.00428
CentroidLinkage	0.662 ± 0.176	1.849 ± 0.921	0.00177 ± 0.00258
WardsMethod	1.829 ± 0.678	1.105 ± 0.869	0.00555 ± 0.00437
DivergenceLinkage	$\mathbf{0.581 \pm 0.201}$	$\mathbf{1.911 \pm 0.932}$	0.00032 ± 0.00071
KMeans	1.764 ± 0.632	1.072 ± 0.833	0.00456 ± 0.00296
KMeans++	1.747 ± 0.612	1.064 ± 0.831	0.00467 ± 0.00302
KMeans-divergence	1.771 ± 0.649	1.089 ± 0.799	0.00506 ± 0.00406

Figure **??** shows the Dunn Index score on each of the real world datasets. Divergence-linkage does not do as well on these datasets: although it does scores best on the WDBC dataset, it is worst on both Wine and Thyroid, and average for the rest. There is no particular clear winner over all the datasets with this measure. Average-linkage does well on WDBC and Thyroid, but badly on Wine, Glass and Iris. Complete-linkage is average throughout, as is Ward's method, and although Centroid-linkage is best on Iris, it average for the rest.

8.2.2.3 *Average Divergence*

Despite being based on the same function, Divergence-linkage is outperformed by Average-linkage on the synthesised datasets, although the difference is slight as shown in Table 6. Single-linkage, again, scores well, and likewise Complete-linkage, Ward's method and KMeans do not; while Centroid-linkage is average.

Figure **??** shows the average divergence of the clusters produced by each of the clustering mechanisms on each of the datasets. Smaller divergences are better. Divergence-linkage has low scores throughout. Ward's method has the highest scores. The Thyroid and Glass datasets are hard for all methods except Divergence-linkage and Average-linkage. Centroid-linkage has a comparably high score for Iris.

Divergence does really well on the internal measures: although not a good as Single-linkage on Dunn and Avg Divergence, it is second best and very close. Ward's

Table 7: 3 classes of iris plant, 50 samples in each, 4 features. Arguably should be 2 clusters since two overlap.

Algorithms	Davies-Bouldin	Dunn	Avg Divergence
SingleLinkage	0.409637	**1.901472**	**0.000034**
AverageLinkage	0.506138	0.533684	0.000092
CompleteLinkage	1.048860	1.239546	0.000129
CentroidLinkage	0.781545	1.711280	0.000358
WardsMethod	0.730370	1.520542	0.000046
DivergenceLinkage	**0.380792**	1.786880	0.000038
KMeans	0.732886	1.539996	0.000055
KMeans++	0.7516854	1.514175	0.000057
KMeans-divergence	0.7736252	1.485374	0.000057

Table 8: Glass:6 classes of glass, 214 samples, distribution: 70 float processed building windows, 76 non-float building windows, 17 float processed vehicle windows, 13 containers, 9 tableware 29 headlamps; 9 features.

Algorithms	Davies-Bouldin	Dunn	Avg Divergence
SingleLinkage	0.489649	**1.858714**	0.000160
AverageLinkage	0.390977	0.068432	**0.000010**
CompleteLinkage	0.670883	1.691238	0.001655
CentroidLinkage	0.601384	1.802741	0.000676
WardsMethod	1.043376	0.728610	0.000692
DivergenceLinkage	**0.332704**	0.875735	0.000015
KMeans	1.409232	0.3262082	0.00074755
KMeans++	1.047465	0.7872234	0.00102929
KMeans-divergence	0.7902541	1.188829	0.00034769

method does well on both types of evaluation. K-Means does not benefit from more spaced out pivots on this dataset.

Divergence again scores best on Davies-Bouldin and is middle-of-the-road for external measures. While single-linkage and average-linkage have reasonable internal scores, they score worst with external. KMeans benefits from divergence.

Divergence scores best on all three internal measures, performs relatively poorly on external measures, single-linkage also performs similarly. Ward's method and complete linkage do well by both types of measure. Divergence helps kmeans on both external and internal measures.

Table 9: Thyroid: 3 classes, 215 samples, distribution: 150 normal, 35 hyper, 30 hypo; 5 features.

Algorithms	Davies-Bouldin	Dunn	Avg Divergence
SingleLinkage	0.133134	5.007502	0.001135
AverageLinkage	1.677889	0.325157	0.000175
CompleteLinkage	**0.000000**	1.325914	0.001141
CentroidLinkage	0.692695	1.838076	0.001781
WardsMethod	**0.000000**	0.923052	0.000864
DivergenceLinkage	**0.000000**	**Infinity**	**-0.000034**
KMeans	1.017782	1.009915	0.00082436
KMeans++	0.9062275	1.157552	0.00112478
KMeans-divergence	0.8428109	1.167863	0.00116933

Table 10: Wine: 3 classes, 178 samples, distribution: 59 class 1, 71 class 2, 48 class 3; 13 features.

Algorithms	Davies-Bouldin	Dunn	Avg Divergence
SingleLinkage	1.015974	0.809457	0.000024
AverageLinkage	**0.453063**	0.548745	**0.000023**
CompleteLinkage	1.036393	1.571030	0.000066
CentroidLinkage	0.751714	**2.160639**	0.000093
WardsMethod	0.845136	2.102532	0.000067
DivergenceLinkage	**0.453063**	0.548745	**0.000023**
KMeans	0.9270488	1.878586	0.00006416
KMeans++	0.9132972	1.957229	0.00006508
KMeans-divergence	0.9310699	1.955152	0.0000664

Table 11: WDBC: 2 classes, 569 samples, distribution: 357 benign, 212 malignant; 30 features

Algorithms	Davies-Bouldin	Dunn	Avg Divergence
SingleLinkage	**0.310634**	**3.219226**	**0.000009**
AverageLinkage	0.364385	2.744352	**0.000009**
CompleteLinkage	1.193467	1.500232	0.000025
CentroidLinkage	0.696482	2.594994	0.000477
WardsMethod	0.864533	2.128549	0.000033
DivergenceLinkage	**0.310634**	**3.219226**	**0.000009**
KMeans	1.055442	1.809522	0.00002371
KMeans++	1.032722	1.858522	0.00002387
KMeans-divergence	1.010494	1.912276	0.0000242

Divergence and average-linkage give the same clustering and are best on two of the three internal measures. The best all round performance come from the centroid based algorithms: centroid-linkage, ward's method and KMeans.

Divergence and single-linkage give the same clustering and are best on all three internal measures. KMeans does best on external measures but not so well on internal measures.

8.2.3 *Seed selection*

Figure **??** shows the impact of increasing the dimensionality of the space on the pivot selection algorithms. First, the number of clusters created is the same as the number of dimensions (Figure **??**). While the max divergence algorithm clearly chooses points that are highly divergent in the lower dimensions, very quickly the algorithm becomes no better than random. Secondly, the cluster size is kept constant (Figure **??**). The effect of the dimensionality is also seen, however, much less pronounced: up to about 100 dimensions, max divergence algorithm produces significantly more divergence seeds.

Next, we looked at the real world datasets with a predefined number of clusters. Table 12 shows the divergence of the seeds: the max divergence algorithm again

Table 12: MSED values of the k chosen seeds averaged over 100 iterations with standard deviations. The synthesised data is the mean of averages for each of the different dimensionalities.

Dataset	KMeans	KMeans++	Max Divergence
Iris	0.0185 ± 0.0146	0.0323 ± 0.0068	**0.0509 ± 0.0037**
Glass	0.0021 ± 0.0013	0.0056 ± 0.0015	**0.0100 ± 0.0002**
Thyroid	0.0132 ± 0.0151	0.0387 ± 0.0218	**0.1029 ± 0.0041**
Wine	0.0041 ± 0.0029	0.0066 ± 0.0034	**0.0171 ± 0.0016**
WDBC	0.0059 ± 0.0062	0.0092 ± 0.0064	**0.0301 ± 0.0055**
synthesised	0.0215 ± 0.0329	0.0507 ± 0.0947	**0.1067 ± 0.1859**

selected seeds with much higher divergence than both random and KMeans++ for all the datasets.

Given that the number of seeds usually chosen for clustering algorithms, like KMeans, is usually quite low, it is clear that max divergence produces much more divergent seeds.

Figure **??** shows how seed selection affects the final outcome of the clusterings, average across all of the real world datasets. There is little to choose between them for the external validity score, F_1, but there is substantial improvement in the internal measures, Davies-Bouldin and Dunn Index.

8.3 CONCLUSION

We have shown a number of uses for the MSED divergence function for clustering tasks. First, Divergence-linkage clustering produces clusters that have low Davies-Bouldin Index scores compared to other clustering techniques. It also produces good scores on the Dunn index and average MSED, too. Clusterings produced in this way are, therefore in general, more tightly packed with greater separation between clusters than other methods. Real world data do not always form tight clusters and may not be easily separable. It does however make a good starting point to examine further MSED based clustering for metric indexing.

Secondly, we demonstrated a selection algorithm to seed KMeans. This algorithm consistently produced seeds that were more divergent than both random selection

and KMeans++. And although this made little difference with external validation, it improved the Davies-Bouldin and Dunn index scores for KMeans. This makes this technique viable as a pivot selection mechanism for pivot table type metric indexes.

9

CONCLUSIONS

BIBLIOGRAPHY