Weights and Measures: An Axiomatic Model for Similarity Computations

Robert K. France
Computing Center
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Abstract:

This paper proposes a formal model for similarity functions, first over arbitrary objects, then over sets and the sorts of weighted sets that are found in text retrieval systems. Using a handful of axioms and constraints, we are able to make statements about the behavior of such functions in reference to set overlap and to noise. The model is then used to analyze, and we hope illuminate, several popular text similarity functions.

0: Introduction.

One of the salient characteristics of advanced information systems is their ability to make approximate matches. This is true of text retrieval systems that produce ranked lists of “best” documents. It is true of relevance feedback systems, and of systems that cluster documents by some heuristic approximation of their similarity. It is also true of fact retrieval or question answering systems that attempt to complete a query statement with the best possible match from a knowledge base. All these systems make use of a metric: a comparison function that rates how well two objects match one another.

Behind this sweeping generalization, of course, lie deep and meaningful differences. Knowledge retrieval and text retrieval systems differ in virtually every important implementation characteristic: data representation, underlying mathematical model, mode of search, and even the similarity computation itself. Text retrieval systems use a wide variety of formulae and algorithms for computing similarity, and the same system may even use different computations in different phases – one for initial search and another for relevance feedback, for instance.

The history of similarity metrics in text retrieval is an exemplar of the scientific method. Different similarity functions and weighting schemes have been proposed, tested, and refined using diligent experimentation and continual sharpening of test collections and evaluation tools. As a result, we have a good body of comparative data to draw on when discussing text similarity functions, with a strong understanding of which works best under a variety of conditions. In addition, this work has lead to considerable refinement in text similarity functions, particularly in the term weighting computations. Nonetheless, it remains true that most of these computations are pragmatically determined, the outcome of a process of finding what works best, rather than the outgrowth of a theoretical model. This limits the discussion, and inhibits transfer to a broader community.

The computations used in the knowledge retrieval community, by contrast, are generally well grounded formally, but virtually undefended pragmatically. For instance, Bayesian logics provide a widely used, if controversial, model for combining uncertainties. Several alternative logics have been proposed [PEAR86] but again the discussion has been centered on formal rather than pragmatic considerations. Information theoretic models have found considerable application in the related area of pattern matching. These quite different calculations have the advantage of having both theoretic and experimental support [RESN89, WATE81], but do not translate easily to the problem of text retrieval.

As information retrieval systems become more widespread and more indispensable, they are being forced to deal with a wider variety of objects. From simple objects composed of a single expanse of text, to composite objects composed of several pieces of text, we are coming into a period where the important objects are digital library objects, involving records for persons, institutions, and complex bibliographic descriptions, as well as increasingly complex text objects. Digital library systems may also handle multimedia and hypermedia objects, which involve parts that are not text at all. To leap successfully into this wider universe, it will be necessary to compute approximate matches among all these sorts of objects. Most importantly, it will be necessary to combine similarity functions and match metrics for objects with dissimilar parts, achieving a figure of merit for the objects as wholes.

Our own experience in the MARIAN library catalog system [FOXF93] indicates that text similarity functions are not always appropriate in matching other sorts of objects. The functions that work best for matching titles, for instance, do not work as well for matching names, and perform poorly for matching composite objects. This is neither disturbing nor entirely unexpected. Names and composites are different sorts of objects, we use different salient characteristics in making intuitive similarity judgments, and so we can expect our computational models to be different as well. What has been more alarming in the MARIAN experience has been how difficult it has been to tune different similarity functions in combination. Deciding which of two authors is the better match to a user's query is straightforward, and calculating which of two titles is the better match to the title portion of a query, while less so, is in well-explored territory. But combining a partial match for an author with an approximate match for a title, in the context of dozens of other bibliographic records with different degrees of similarity in those two fields, in hopes of producing a sensibly ranked list for the user to peruse – that has proven a delicate and formidable problem.

Pragmatics alone will not be enough to solve the problems in this wider universe of objects. We need a mathematical model for describing and analyzing different similarity functions, so that we may combine and tune them. This model should be fully abstract, not rooting in either a particularly family of functions or a particular data model. This paper makes several steps in that direction – some confident, others tentative – in hopes of sparking further discussion.

1. Similarity and Difference.

We begin this process by presenting a formal model of a similarity function, and work from there to some initial axioms for the function's behavior. Since we want to discuss a broad range of functions, we will keep the initial discussion abstract. In particular, in this section we will not put any requirements on the objects being compared, except the intuitive requirement that it make sense to describe some of them as being "similar" to others. Without presumptions about the nature of the compared objects, we can say little about the detailed behavior of a similarity function over different objects. In later sections, as we will refine our picture of the objects being compared, we will be able to say progressively more.

When we say that objects are similar, we have in mind something more than the Boolean comparison of equality. We are interested in base classes of objects in which we can find pairs of objects that are neither identical not totally disparate, but where one object is somehow like the other. A similarity function will then be a (better or worse) formalization of that likeness. A base class can support a number of different similarity functions, both because some functions will capture our naive judgments of likeness better and because there may be several different standards by which objects in the base class may be judged to be similar. We will require, however, that the results of a similarity comparison be themselves comparable, so that we may say that one match is better or worse than another. This is a more stringent condition than simply requiring that the function order the objects of the base class by similarity to any given query object. Rather, we need to be able to make comparisons on the results of similarity calculations so that we can say, for example, that object **a** is a better match to query **q** than object **b** is to query **p**.

On the other hand, the requirement is less stringent than requiring that the results of similarity functions be quantifiable, in the sense of being able to be mapped into some number system. Quantifiable measures will certainly fill the bill, but are in fact stronger than needed. In fact, the structure that provides exactly as much as we need is a total order.

Definition 1.1: A *partial ordered set (poset)* is a set $\mathbb{A}$ together with a relation $\leq$ such that for all a,b,c $\mathbb{A}$:

- $a \leq a$ (reflexivity)
- if $a \leq b$ and $b \leq c$, then $a \leq c$ (transitivity)
- if $a \leq b$ and $b \leq a$, then $a = b$ (antisymmetry).

A *totally ordered set* is a poset for which also for all a,b $\mathbb{A}$

- either $a \leq b$ or $b \leq a$.

Definition 1.2: A *weight* is a totally ordered set with both a least element (bottom) and a greatest element (top), here denoted $\mathbb{0}$ and $\mathbb{1}$ respectively, for reasons that will later become clear.

All of the familiar number systems are partial orders, and most are total orders (the complex numbers are not).

Examples of total orders with top and bottom include the real numbers between 0 and 1, and $\mathbb{Z}^{+}$, the integers together with the "extra" elements $+\infty$ and $-\infty$. We will use the top element to represent a perfect match, and the bottom to represent a complete miss.

Definition 1.3: A *similarity function* sim(a, b) on a base class $\mathbb{B}$ is a map that takes two elements of $\mathbb{B}$ and returns a member of a weight class $\mathbb{W}$ such that

Axiom 1.4: sim(a, a) = $\mathbb{1}$ for all a $\mathbb{B}$.

A *reflexive* similarity function is one for which also

sim(a, b) = sim(b, a).

If sim(a, b) is not reflexive, we will call it a *directed* similarity function and denote it

sim(a, b).

When discussing directed similarity functions, we will sometimes refer to the first argument as the *query* and the second as the *target* object.

Both reflexive and directed similarity functions occur in the literature and have been used successfully in retrieval systems. The pure vector cosine function, for instance, is reflexive, but quorum match, which ranks all targets with **k** query terms higher than any with (**k**-1) terms, is not. Neither are any of the hybrid inner product functions in [SABU88] where the computations used to weight query terms differ from those used to weight terms from the target document. Outside of text retrieval, Bayesian combination is reflexive, but many pattern match calculations are directed. Which type of similarity function is more desirable appears to depend on the application.

When two objects of the base class are totally dissimilar, or have nothing in common, a similarity function should return $\mathbb{0}$. This is the intended use for the bottom element. We cannot make this an axiom, however, without defining what it means to have nothing in common, for which we need some presumptions about the structure of the base class. Instead, we will call this

Constraint 1.5: sim(a, b) = $\mathbb{0}$ whenever a and b have nothing in common.

A similarity function is presumed to be total on $\mathbb{B}\times\mathbb{B}$. That is, sim(a, b) must produce a valid weight for any a,b $\mathbb{B}$. In practice, many similarity functions return $\mathbb{0}$ for most pairs. On the other hand, it is not necessary for a similarity function to cover the weight class. The *identity* map, for instance, defined by

$$\mathrm{sim}_{id}(a, b) = \begin{cases} \mathbb{1} & \text{if } a = b \\ \mathbb{0} & \text{if } a \neq b \end{cases}$$

is an uninteresting, but perfectly valid, similarity function. Nor is a similarity function required to be one-to-one.

Many similarity functions will return the same value for distinct pairs of base elements, judging them to be equally similar. In particular, some similarity functions will judge distinct elements of the base class to be completely similar, and thus map distinct elements to $\mathbb{1}$. Thus although the weight class is a total order, and although a similarity function induces an ordering on the elements of the base class by virtue of that order, the ordering that it induces is not antisymmetric and thus not a partial order, but rather what might be referred to as a total preorder.

Reflexive similarity functions have a certain kinship with distance metrics. A distance metric is also a function from pairs to elements of some base class into a total order with bottom: in this case the real number interval $[0, +\infty)$. If we extend the interval to include the point $+\infty$, it qualifies as a weight under the standard $\leq$ ordering, but a weight that comes with some important added structure.

Definition 1.6: An *extended distance metric* on a base class $\mathbb{B}$ is a map d: $\mathbb{B}\times\mathbb{B} \rightarrow [0, +\infty]$ such that:

- $d(a, a) = 0$
- $d(a, b) > 0$ whenever $a\neq b$
- $d(a, b) = d(b, a)$ (symmetry)
- $d(a, b) + d(b, c) \geq d(a, c)$ (triangle inequality)

There is a certain inverse symmetry between the concepts of distance and similarity. As the distance between two objects grows, their similarity can be said to decrease, and so forth. And we can define a simple bijection from $[0, +\infty]$ to $[0, 1]$ using the functions $x=e^{-y}$ and $y=-\ln(x)$ that identifies a distance of 0 with a similarity of $\mathbb{1}$ and an infinite distance with a similarity of $\mathbb{0}$ in a very satisfying way. Under the bijection addition of distances is isomorphic to multiplication of similarities, and the arithmetic $\geq$ corresponds to $\leq$ in the weight class. Unfortunately after this the analogy fails, as there are many valid similarity functions for which nothing like the triangle inequality holds. For instance, let **a** and **c** be texts that have no words in common, and let **b** be a text that shares words with both **a** and **c**. Then using any standard text similarity metric, including quorum, cosine, or EMIM:

$\mathrm{sim}(\mathbf{a}, \mathbf{c}) = \mathbb{0}$	($d(\mathbf{a}, \mathbf{c}) = +\infty$, OK)
$\mathrm{sim}(\mathbf{a}, \mathbf{a}) = \mathbb{1}$	($d(\mathbf{a}, \mathbf{a}) = 0$, OK)
$\mathrm{sim}(\mathbf{a}, \mathbf{b}), \mathrm{sim}(\mathbf{b}, \mathbf{a}), \mathrm{sim}(\mathbf{b}, \mathbf{c}), \mathrm{sim}(\mathbf{b}, \mathbf{c}) > \mathbb{0}$	($d(...) < +\infty$, OK)
$\mathrm{sim}(\mathbf{a}, \mathbf{b})\cdot\mathrm{sim}(\mathbf{b}, \mathbf{c}) \leq \mathrm{sim}(\mathbf{a}, \mathbf{c})$	($d(\mathbf{a}, \mathbf{b}) + d(\mathbf{b}, \mathbf{c}) \geq d(\mathbf{a}, \mathbf{c})$, OK)
BUT	
$\mathrm{sim}(\mathbf{a}, \mathbf{b})\cdot\mathrm{sim}(\mathbf{b}, \mathbf{a}) \leq \mathrm{sim}(\mathbf{a}, \mathbf{a})$	(not OK).

This situation cannot be put right by changing the direction of the inequality or by playing with the operator. It is necessary and proper that for these three objects, both of the last two statements hold: each of the partial similarities (and thus their sum and product) must be greater than the total dissimilarity that holds between **a** and **c**, but even their sum may not be greater than the condition of total similarity that holds between **a** and itself.

2. Texts and Sets

A text can be considered to be a sequence of tokens, each of a certain type. Automatic processing systems perform various transformations on the tokens; ignoring white space and punctuation, for instance, or mapping all inflectional transformations of a word to the root form. These affect the performance of the system, but not the model: the result is still a sequence of tokens drawn with repetition from a universe of types. A more radical transformation, commonly used for both formal modeling and tractable processing of text, is to forget the order of the sequence. In this abstraction, a text is considered to be a bag or multi-set, where unique elements can occur multiple times. In general, no relations except equality and inequality hold between the elements, which are generally referred to as *terms*.

From multi-sets it is a small step to sets where each term is associated with a count of the number of times it appears in the text. These common representations motivates our choice of sets and weighted object sets as base classes for similarity functions.

Set classes need no introduction. Sets are the familiar objects of mathematics; we will lean to computer science so much as to say that a set *class* presupposes a specific underlying element class. This underlying class could be simply the class of all objects (*pace* arguments about self-referentiality), or it could be restricted, for example to terms from a given text collection. When it is restricted, the element class may support operations and similarity functions of its own; for now we will pass over this complexity and assume nothing more than that it supports an identity function.

Weighted object sets are sets where each element has an associated weight. These occur frequently in text systems: they can be used to represent texts, as mentioned above, or sets of ranked retrieval results. In multi-leveled systems like MARIAN, the results set of one approximate match become the material for further similarity computations, and are used recursively.

Definition 2.1: A *weighted object set* is a set $\mathbf{s}$ together with a function wt($\mathbf{s}$, $\mathbf{e}$) that returns a weight other than $\mathbb{0}$ for every $\mathbf{e}$ $\mathbf{s}$.

This definition is semantically equivalent to requiring all the elements of $\mathbf{s}$ to be ordered pairs of object and weight, but will be simpler for our purposes. There is an obvious correspondence between weighted sets and fuzzy sets; other similar or equivalent structures will no doubt occur to each reader.

3. Set Similarity Functions.

Similarity functions for sets can draw on three levels of information:

- *the underlying element class.* The elements that make up the sets being compared may themselves support one or many similarity functions. Which we choose and how it functions will affect the similarity function at the set level.
- *the composition of the sets being compared.* The size of the overlap between the sets is always important to the semantics of a set similarity function. For some functions, the set difference in one or both directions – unmatched elements in the query or noise in the target – is also important. Finally, which set elements fall within the overlap may be important, in cases where it is more important to match certain elements than others. This is reflected in the following considerations.
- *global characteristics of the set class.* Since a similarity function is defined across an entire class of sets, it may make use of information about the class as a whole. An IDF function over a text collection, for instance, assigns more or less importance to the terms in a comparison based on their global frequency in the collection. Other global characteristics include sizes, maxima, and the very semantics of the class. An example of the last is any similarity function for composite documents that emphasizes words occurring in a title over those occurring in the bulk of the text.

Functions for weighted object sets may also draw on:

- *local weights within the sets being compared.* If certain elements in one or both of the sets are weighted more highly than others, a similarity function may give precedence to matches where those elements fall into the overlap. Any of the family of TF text similarity functions does this, preferring matches by more frequent terms over less frequent. This can occur either separately or in combination with global weighting.

When we restrict our attention to base classes whose members are sets, we can say a great deal more about how similarity functions should behave. For one thing, we now know something about the composition of our base class members, and so can make some statements about how that composition affects the similarity function. For instance, we can sharpen Constraint 1.5 to a formal statement:

Axiom 3.1: For any similarity function sim($\mathbf{a}$, $\mathbf{b}$) over a set class $\mathbb{S}$, sim($\mathbf{a}$, $\mathbf{b}$) = $\mathbb{0}$ when and only when $\mathbf{a}$ and $\mathbf{b}$ have no elements in common, that is,

sim($\mathbf{a}$, $\mathbf{b}$) = $\mathbb{0}$ when and only when $\mathbf{a} \cap \mathbf{b} =$.

In particular, this implies that sim($\mathbf{a}$,) = sim(, $\mathbf{b}$) = $\mathbb{0}$, for both reflexive and directed set similarity functions.

As important as it is to be able to discuss the composition of base objects, the operations of the class are even more useful. In particular, we can focus on union and intersection. If $\mathbb{S}$ is a set class, and sim($\mathbf{a}$, $\mathbf{b}$) is a similarity function on $\mathbb{S}$, then we can state with confidence

Axiom 3.2: sim(**a**, **a**∩**b**) ≤ sim(**a**, **a**) = $\mathbb{1}$, with equality when and only when **b**   **a**.

Taking something away from **a** gives us a set object that is no longer identical to **a**, so it is reasonable to expect a similarity function to return a value less than $\mathbb{1}$ when **a** is diminished. The implications of this are subtle. The axiom does imply that taking progressively more from **a** results in sets that are progressively less similar to **a**. In other words, a function obeying Axiom 3.2 must produce monotonically decreasing weight values over any chain of subsets of the query, from $\mathbb{1}$ when the target is equal to the query, through smaller and smaller values as it is progressively subsetted, to $\mathbb{0}$ for the empty target in accordance with Axiom 3.1. Axiom 3.2 does *not* imply, however, that taking something away from an arbitrary target set must decrease its similarity to the query. As a simple example, taking away noise from some superset **c** of **a** may result in a closer match to **a**. In fact, taking away *enough* noise from **c** may result in so much improvement as to offset a decrease in the overlap of **c** and **a**, so that while we will state without proof that:

Proposition 3.3: sim(**a**, **b**∩**c**) ≤ sim(**a**, **b**) when **a**   **c**,

we cannot even strengthen the proposition to a "when and only when," and we will make no predictions at all about arbitrary intersections.

When sim(**a**, **b**) is reflexive, Axiom 3.2 also implies:

Theorem 3.4: If sim(**a**, **b**) is a reflexive set similarity function, then

(i) sim(**a**∩**b**, **a**) ≤ sim(**a**, **a**) = $\mathbb{1}$, with equality when and only when **b**   **a**.

(ii) sim(**a**, **a**   **b**) ≤ sim(**a**, **a**) = $\mathbb{1}$, with equality when and only when **a**   **b**.

Proof: (i) is immediate from the definition of a reflexive similarity function. For (ii), note that

sim(**a**, **a**) = $\mathbb{1}$ = sim(**a**   **b**, **a**   **b**)

≥ sim((**a**   **b**)∩**a**, **a**   **b**) by (i)

= sim(**a**, **a**   **b**) by absorption.

3.4(ii) states that adding irrelevant things to **a** also results in something that is on longer identical to **b**. A reflexive similarity function must be sensitive to noise. This is not implied for directed similarity functions. Directed similarity functions may either take 3.4(ii) as an axiom, or ignore it without fear of contradiction. Salton and Buckley's hybrids, for instance, are sensitive to unmatched terms in both query and target, but exact much higher penalties for unmatched query terms than for target noise. Quorum functions and certain frame match functions, in contrast, are impervious to noise in the target.

Given a (reflexive or directed) similarity function that obeys 3.4(ii), and a particular query **q**, we can examine the structure of weights generated by sim(**q**, **s**) for **q** **s**. Call this unary weighting function ω(**s**). In the topology *Q* of subsets of **q**, we have already established that ω(**t**) ≥ ω(**s**) whenever **t** **s**. We have not established any comparison principles for **s** and **t** when **s** and **t** are distinct subsets of **q**. However, if we accept

Constraint 3.5: Where the weight class 𝕎 is (a subset of) a ring with operator having 𝟘 as the identity,

sim(**q**, **s** **t**) = sim(**q**, **s**) sim(**q**, **t**) whenever **s** ∩ **t** =

then ω(**s** **t**) = ω(**s**) ω(**t**) whenever **s** ∩ **t** = , and the structure <**q**, *Q*, ω> is a measure space (see, e.g., [HALM74]). In particular, if we choose 𝕎 to be the real interval [0, 1], set 𝟘 to 0 and 𝟙 to 1, and restrict our attention to finite subsets of **q**, <**q**, *Q*, ω> is a probability space, and the familiar laws of probability theory apply.

Since the subset topology is (isomorphic to) a Boolean algebra, the element **q'** ("**q**-inverse," defined as all the elements in the universe except those in **q**) is well-defined and unique. The subset lattice between and **q'** defines a topology of noise in the context of **q**, on which ω also defines a measure space. Now, it is no restriction to assume that a query **q** will be finite, since it comes from either a user or a finite text collection, but we may not want to put any such limitations on **q'**. Nonetheless, since any member of the noise topology that we care about is the result of abstracting **q** from a finite document, we can be happy to restrict ourselves to the topology *Q'* of finite subsets of **q**, which is exactly the topology required for <**q'**, *Q'*, ω> to be a probability space. Whether and how the two spaces are combined determines completely the relationship between overlap and noise, and thus can be viewed as a specification of the parent similarity function.

Not all similarity functions are set-theoretic. Not all make use of a ring for weights. Importantly, not all that do so use the ring operators in a way that respects the set operations, and thus satisfies Constraint 3.5. But for those that do, we can specify semantics precisely using the strictures of measure theory. Furthermore, we can define a set similarity function, perhaps uniquely, by stating whether and how the overlap measure and the noise measure combine.

4. Examples and Analysis.

To show how these axioms work in practice, this section provides a quick analysis of two common families of text retrieval functions: quorum match and vector cosine functions. Both these functions use the real interval [0, 1] as their weight class, and make use of both addition and multiplication. Since the interval is not closed under addition, the first thing that we must verify is that the functions in fact never do return a value out of this range.

The first family are those we have been calling *quorum* similarity functions. This name has some currency in the text retrieval community, and the family has considerably more. Quorum functions are directed, giving precedence to the query set and ignoring noise in the target set. Examples of functions in the family are:

$$\text{quorum}_{card}(\mathbf{q}, \mathbf{t}) = |(\mathbf{q} \cap \mathbf{t})| \;/\; |\mathbf{q}|$$

$$\text{quorum}_{avg}(\mathbf{q}, \mathbf{t}) = \sum_{e\ \ q} \left[\text{wt}(\mathbf{q}, \mathbf{e}) \cdot \text{wt}(\mathbf{t}, \mathbf{e})\right] \;/\; \sum_{e\ \ q} \text{wt}(\mathbf{q}, \mathbf{e})$$

$$\text{quorum}_{scale}(\mathbf{q}, \mathbf{t}) = \sum_{e\ \ q} \left[\text{wt}(\mathbf{q}, \mathbf{e}) \cdot \text{wt}(\mathbf{t}, \mathbf{e})\right] \;/\; \sum_{e\ \ q} \text{wt}(\mathbf{q}, \mathbf{e})^2$$

Of these, we will note that quorum$_{avg}$ violates Axiom 1.4, since

$$\text{quorum}_{avg}(\mathbf{q}, \mathbf{q}) = \sum \text{wt}(\mathbf{q}, \mathbf{e})^2 \;/\; \sum \text{wt}(\mathbf{q}, \mathbf{e}) < 1$$

unless wt($\mathbf{q}$, $\mathbf{e}$) = 1 for all $\mathbf{e}$ $\quad$ $\mathbf{q}$. But if wt($\mathbf{q}$, $\mathbf{e}$) = 1 for every element in every set, then the function collapses into quorum$_{card}$.[1] On the other hand, quorum$_{scale}$, which satisfies all our axioms, may produce values in excess of 1 when wt($\mathbf{t}$, $\mathbf{e}$) = 1 for all $\mathbf{e}$ $\quad$ $\mathbf{q}$. Thus leaves only quorum$_{card}$, which is the function specified by using only the query overlap measure space and ignoring the noise measure space.

The second family of functions are the symmetric similarity functions based on the vector cosine function. Vector cosine should be familiar to members of the retrieval community, so we will dispense with development of the underlying model. We will note that the weight-valued vectors – for the purpose of this discussion vectors with their real components in the interval [0, 1] – can be mapped precisely onto the finite weighted object sets, and vice versa.

The vector cosine function is so defined that its range is in [-1, 1]. For weight-valued vectors, this is restricted to [0, 1] as desired. So we need fear no ill-definedness. Further, any pure vector cosine function satisfies Axiom 1.4 by design. Such a function also satisfies 3.1 through 3.4(ii). On the other hand, it does not satisfy Constraint 3.5. For instance, take a query $\mathbf{q}$ and divide it into two parts, $\mathbf{q_1}$ and $\mathbf{q_2}$ so that $\mathbf{q_1} \quad \mathbf{q_2} = \mathbf{q}$ and $\mathbf{q_1} \cap \mathbf{q_2} = \quad$. We note that the inner product $<\mathbf{q}, \mathbf{q_1}> = \sum \text{wt}(\mathbf{q}, \mathbf{e}) \cdot \text{wt}(\mathbf{q_1}, \mathbf{e}) = \sum \text{wt}(\mathbf{q_1}, \mathbf{e})^2 = ||\mathbf{q_1}||^2$. So

$$\text{sim}_{cos}(\mathbf{q}, \mathbf{q_1}) = ||\mathbf{q_1}|| \;/\; ||\mathbf{q}||.$$

Similarly, $\text{sim}_{cos}(\mathbf{q}, \mathbf{q_2}) = ||\mathbf{q_2}|| \;/\; ||\mathbf{q}||$. By definition $\text{sim}_{cos}(\mathbf{q}, \mathbf{q}) = 1 = ||\mathbf{q}|| \;/\; ||\mathbf{q}||$. Therefore the only way that Constraint 3.5 can be satisfied is if

$$||\mathbf{q_1}|| + ||\mathbf{q_2}|| = ||\mathbf{q}||$$

$$\sqrt{\sum \text{wt}(\mathbf{q_1}, \mathbf{e})^2} + \sqrt{\sum \text{wt}(\mathbf{q_2}, \mathbf{e})^2} = \sqrt{\sum \text{wt}(\mathbf{q}, \mathbf{e})^2}$$

Squaring both sides, we get:

$$\sum \text{wt}(\mathbf{q_1}, \mathbf{e})^2 + 2\sqrt{\sum \text{wt}(\mathbf{q_2}, \mathbf{e})^2} \cdot \sqrt{\sum \text{wt}(\mathbf{q_1}, \mathbf{e})^2} + \sum \text{wt}(\mathbf{q_2}, \mathbf{e})^2 = \sum \text{wt}(\mathbf{q}, \mathbf{e})^2$$

$$2\sqrt{\sum \text{wt}(\mathbf{q_2}, \mathbf{e})^2 \cdot \sum \text{wt}(\mathbf{q_1}, \mathbf{e})^2} = 0$$

which can only be true when either $\mathbf{q_1}$ or $\mathbf{q_2} = \mathbf{0}$.

This lack of additivity is a well-known problem with vector cosine. Known by several names, the problem manifests itself in composite document retrieval and in retrieval over underlying classes where partial matches, as well as exact matches, are possible. In the first case it produces inconsistencies in the similarities of two composite documents considered on the one hand as vectors of fields, each of which is a vector of terms, and on the other as flat vectors of ordered <field, term> pairs. The second case can arise, for instance, in a lexically sophisticated system where two or more terms can be taken as partial matches to the same root. Then the cosine similarity of the vector with the terms considered separately is different from that obtained if they are collapsed into the root. Here the problem arises in the abstract, manifest as a violation of the measure space constraints.

Since the lack of additivity shown by vector cosine is clearly related to the use of vector norms for scaling, we briefly consider two popular unscaled inner product metrics studied by Bruce Croft [CROF83] and Donna Harmon [HACA90]. Both Croft's metric,

$$\text{wt}_{\text{Croft}}(\mathbf{t}, \mathbf{e}) = \alpha + (1-\alpha)\cdot\text{ct}(\mathbf{t}, \mathbf{e})/\text{maxCt}(\mathbf{t})$$

$$\text{sim}_{\text{Croft}}(\mathbf{q}, \mathbf{t}) = \sum_{\mathbf{e} \in \mathbf{q}} \left[(\gamma + \text{IDF}(\mathbf{e})) \cdot \text{wt}_{\text{Croft}}(\mathbf{t}, \mathbf{e}) \right]$$

where α and γ are constants, IDF($\mathbf{e}$) is a global weight on $\mathbf{e}$, ct($\mathbf{t}$, $\mathbf{e}$) is the number of times $\mathbf{e}$ appears in $\mathbf{t}$, and maxCt($\mathbf{t}$) is the maximum count of any element of $\mathbf{t}$; and Harmon's

$$\text{wt}_{\text{Harmon}}(\mathbf{t}, \mathbf{e}) = \log(\text{ct}(\mathbf{t}, \mathbf{e})+1) / \log(\sum_{\mathbf{f} \in \mathbf{t}} \text{ct}(\mathbf{t}, \mathbf{f}))$$

$$\text{sim}_{\text{Harmon}}(\mathbf{q}, \mathbf{t}) = \sum_{\mathbf{e} \in \mathbf{q}} \left[\text{IDF}(\mathbf{e})\cdot\text{wt}_{\text{Harmon}}(\mathbf{t}, \mathbf{e}) \right]$$

do obey Constraint 3.5. They both share the problem observed in Note 1: query weights do not enter explicitly into the calculations, inducing a disparity in the treatment of queries and documents. In particular, this means that they violate Axiom 1.4 for any $\mathbf{q}$ with weights other than 1. Either function could be patched to avoid this disparity by adding a wt($\mathbf{q}$, $\mathbf{e}$) to the inner product, but they would still evince a deeper problem: while the wt($\mathbf{t}$, $\mathbf{e}$) functions do produce values in the interval [0, 1], the sim($\mathbf{q}$, $\mathbf{t}$) functions produce values that vary with the length and IDF values of the query. Again, they could be patched by scaling them by query length, but only by dint of radical changes. Both functions appear to work well in their original form, and we would not want to alter them without further experimentation.

Other than that, both inner products follow Axioms 3.1 and 3.2. Harmon’s measure also follows 3.4(ii). Adding noise to **t** causes $\sum$ct(**t**, **f**), the total number of terms in **t**, to increase, thus decreasing all the weights of the matching terms. Croft’s measure does not explicitly follow 3.4(ii), but may evince the same effect. The term weights in Croft’s measure vary depending on the maximal count of any term in the text. Adding additional terms with counts lower than the existing maximum will have no effect on the measure, but adding a new maximum will cause all existing term weights, and the overall similarity, to decline. Some objects in the topology of noise will cause this to happen; others will not, and describing which are which appears beyond the capacity of the current model.

5: Conclusion.

We have presented an axiomatic model for similarity functions, with emphasis on similarity functions over sets and weighted sets. This model is not intended to be a final word – in particular, we are not sure that Constraint 3.5 is necessary – but like all models is target for revision, reinterpretation, and replacement.

Even in this small a beginning we have been able to realize some useful results. these include:

- that similarity functions are not in any sense inverted distance metrics,
- that all set similarity functinos depend monotonically on the size of the overlap between query and target, and in addition
- that symmetric set similarity functions must be sensitive to noise in both sets.

In addition, we have been able to point precisely to several problems of existing similarity functions. the two most common are *incomparability,* where the results of one comparison cannot be compared with another, and *irreflexivity,* where objects are not accounted perfect matches to themselves. In all fairness, it must be said that most of the incomparable functinos discussed were created for use in systems where each retrieval is a separate action, and comparison among retrieval results never occurs. When systems do combine results from several queries, however, or must retrieve compound or recursively defined objects, incomparability becomes a major problem.

A tantalizing result of this discussion id the identification of set similarity functinos with measures. If this result is accepted, it provides us with a powerful framework for analyzing, specifying, and verifying conforming similarity functions. We are uncertain, though, how wide a subset of interesting functions conform to the strictures of measure theory. Certainly several interesting and useful text retrieval functinos do not.

Much remains to be accomplished. I one direction, we are prepared to extend our analysis to specific TF and IDF weighting functions on the one hand, and to information-theoretic similarity functinos like cross-entropy and EMIM on the other. At the same time, there is the question of function behavior. The axioms given here apply to similarity functinos that drop off quickly, slowly, or linearly in a variety of factors. Our experience in MARIAN leads us to believe that combining similarity functinos of different shape makes the combined function ill-behaved. Therefore this strikes us as an important direction to explore.

As stated in the introduction, this formalization has only been possible because of the experimental exploration that has gone before. In this, we are particularly indebted to the comparative results of Gerard Salton’s group, as well as results by C.J. van Rijsbergen, Bruce Croft and Karen Sparck Jones. As well as the results cited throughout the paper, we have found invaluable the summaries of Harmon [HARM92] and of Jones and Furnas [JOFU87]. Conversations with Ed Fox have improved this work no end; needless to say, any flaws remaining are the sole property of the author. IT is the dedication of people like these to keeping the science in computer science that make possible any meaningful formalism at all.

Notes

1. We could escape this by partitioning our supporting set class between queries, all of which have top weights for all elements, and targets, which may have arbitrary weights. Many retrieval systems do appear to do this, and the common practice of users to create queries like “famous,women,American,history” supports it. We find it unsatisfying, though, not only because it multiplies categories unnecessarily, but because it limits the function’s effectiveness for retrieval. Making queries and targets separate kinds of entities means that the similarity function cannot be used to compare targets, as for clustering, and that we cannot use targets as queries, as in relevance feedback. It can be objected that we can always convert a target object into a query object by forcing all its weights to $\mathbb{1}$, but this is unsatisfying in a different way: it throws away precisely the distributional information on which most approximate matches are based.

Bibliography

[CROF83] Croft, W. Bruce. "Experiments with representation in a document retrieval system." *Information Technology: Research and Development* **2**:1, 1983, pp. 1-21.

[FOXF93] Fox, Edward A., Robert K. France, Eskinder Sahle, Amjad Daoud and Ben E. Cline. "Development of a modern OPAC: from REVTOLC to MARIAN." *Proc. of the 16th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval.* ACM, 1993, pp. 248-259.

[HALM74] Halmos, Paul R. *Measure Theory.* New York: Springer-Verlag, 1974.

[HACA90] Harmon, Donna and G. Candela. "Retrieving records from a gigabyte of text on a minicomputer using statistical ranking." *JASIS* **41**:8, 1990, pp. 581-589.

[HARM92] Harmon, Donna. "Ranking algorithms." in Frakes, W.B. and R. Baeza-Yates, Eds. *Information Retrieval.* Englewood Cliffs: Prentice-Hall, 1992, pp. 363-392.

[JOFU87] Jones, William P. and George W. Furnas. "Pictures of relevance: a geometric analysis of similarity measures." *JASIS* **38**:6, 1987, pp. 420-442.

[PEAR86] Pearl, Judea. "Fusion, propagation, and structuring in belief networks." *Artificial Intelligence,* **29**:3, 1986, pp. 241-288.

[RESN89] Resnikoff, Howard L. *The Illusion of Reality*. New York: Springer-Verlag, 1989.

[SABU88] Salton, Gerard and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." *Information Processing & Management* **24**:5, 1988, pp. 513-523.

[SAMC83] Salton, Gerard and McGill, M. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill, 1983.

[SPAR79] Sparck Jones, Karen. "Experiments in relevance weighting of search terms." *Information Processing & Management,* **15**:3 (1979), pp. 133-144.

[VANR79] van Rijsbergen, C.J. *Information Retrieval,* 2nd Ed. London: Butterworths, 1979.

[WATE81] Watenabe, S. "Pattern recognition as a quest for minimum entropy." *Pattern Recognition* **13** (1981), pp. 381-387.