When Stopping Rules Don't Stop

Robert France
Computing Center
Virginia Polytechnic and State University
Blacksburg, VA 24061
france@vtopus.cs.vt.edu

Abstract

Performing ranked retrieval on large document collections can be slow. The method of *stopping rules* has been proposed to make it more efficient. Stopping rules, which terminate search when the highest ranked documents have been determined to some degree of likelihood, are attractive and have proven useful in clustering, but have not worked well in retrieval experiments. This paper presents a statistical analysis of why they have failed and where they can be expected to continue failing.

Ranked retrieval systems, particularly those based on approximate matching functions, have many attractive features. They are easy to use, as queries can be expressed as simple lists of terms. They are tolerant of mistakes and imprecision in queries, and they do not require users to understand the operation of the underlying system. They also provide a good interface to large document collections, where even the most carefully considered Boolean queries may retrieve thousand of documents. Ranked retrieval systems may also retrieve thousands of documents or even tens of thousands, but they can present only the top few documents and avoid overwhelming the user.

The key to this behavior is some heuristic weighting function that ranks documents in the collection by their similarity or relevance to the user's query. A number of these functions have been proposed over the years [SALT&BUCK88, ROBERTSON??, HARMON92], none of which has proven to be best in all cases. Different types of queries, different types of document collections, and even different statistical patterns among the documents in the collection all have an impact on which weighting function works best. A common feature of the functions proposed, however, is that they are typically summative in nature. The overall figure of merit for the relevance of a document to a query is expressed as a sum over the parts of the query, be those document attributes, clauses in an extended Boolean representation, or simply words in a text expression. This is a problem.

The most common types of ranked retrieval search engines use an inverted file structure. The components of a document are collected and indexed as *posting lists* of document-weight pairs retrievable by component type.[*] If the relevance of a document to a query is to be determined by a sum over all components in the query, then the exact rank of a document depends of its weights in all the posting lists. Computing exact weights for all matching documents requires touching every item in every posting list at least once. Many posting lists are quite long. In a collection of a million documents, the most common terms in the language can have posting lists with hundreds of thousands of elements. Computing exact weights can require many operations, and most importantly many disk accesses.

This problem can be attacked on at least two fronts: a search engine can sacrifice some precision in final weights, and thus in final ranks, or it can avoid calculating some fraction of the text weights. Both these strategies play into the method of *stopping rules* first proposed by Smeaton and van Rijsbergen [SM&vRIJS81] in the context of finding the nearest neighbors to a text document. In outline, the idea behind the method is to examine first those lists and postings that will have the most effect on the final weights of any document in the result set, and to apply some sort of heuristic rules periodically throughout the search in hopes of being able to stop before the low-weight lists or postings have been examined. The method was refined and applied to the problem of ranked retrieval by Buckley and Lewit [BUCK&LEW85], but with disappointing results. They found in experiments on standard collections that only "drastic" sacrifices in the dependability of final weights produced a notable improvement in performance. Further refinements of the method were made by Wong and Lee [WONG&LEE91, LEE&WONG91], who also found the performance disappointing. Harmon and Candela [HARM&CAND90] used a related method to prune search results after all postings had been examined. Their improvements were greater, but still less than might have been expected.

***** For the sake of simplicity, we will restrict our discussion here to the problem of matching a single piece of query text to a single piece of document text. In this case, the document components are the terms that make up the text, and posting lists of <documentID, weight> pairs are associated with individual terms. The same considerations apply when document components include names, dates, structured texts, and so forth. Extended Boolean systems raise further problems, but the ones that we examine are still pertinent.

Harmon and Candela's experiments differed from both Buckley and Lewit's and Wong and Lee's in that they were made on a large collection of real-world documents. Encouraged by their results, and bolstered by our own further refinement in the stopping conditions, we decided to include an opportunistic search engine in the MARIAN system.* Unfortunately, our experience in MARIAN searches only confirmed the experiments of Buckley, Lewit, Wong and Lee. In most cases, searches either could not be stopped short of complete posting list exploration, or could only be stopped when the vast majority of the postings had been examined. The search engine could only be persuaded to stop significantly before the end of the search by setting the acceptable error in weights so low that an initial result segment of 20 documents was virtually guaranteed to miss at least one document that should have been there.

Such poor performance is all the more mystifying since stopping rules have been shown to perform well in their original context, that of clustering documents into nearest-neighbor groups [LUCARELLA88, RASMUSSEN92]. This paper attempts to explain why stopping rules work less well in the context of ranked retrieval. We first present a model for the inverted file structure of a text collection and the retrieval process, together with supporting evidence from the document collection used in MARIAN. We then introduce our opportunistic search method, together with our refined stopping rules. The final section is a statistical analysis of when the stopping rules can legitimately terminate a search before all postings have been examined.

1. Model

The individual posting lists in an inverted file system and the results of a ranked retrieval have the same formal structure. Both are examples of *weighted object sets:* sets of objects, each of which has an associated weight. In this paper we will take *weights* to be real numbers in the interval [0,1], where 0 means least interesting and 1 most interesting. As it happens, zero

* MARIAN [FOXetal93] is a ranked retrieval library catalog system developed at the Virginia Tech Computing Center. Its purpose is to provide an alternative public catalog for easy network access where approximate matching methods serve as an alternative to more rigid access points. MARIAN is implemented as a set of clients communicating with a distributed server. The server includes both exhaustive and opportunistic search engines, as well as several other innovative features. The search results in this paper are drawn from our first full-size database: a 1994 snapshot of the full Virginia Tech library catalog of 960,000 MARC records. Records are indexed by title; personal, corporate, or conference author; subject; and notes. The posting lists shown are from the title index.

weights never occur in weighted object sets, but it is useful to have the value in the model. We purposely avoid defining objects. In actual implementations, set elements tend to be object surrogates such as ID numbers anyway; thus both the implementation and model make no commitment on objects beyond identity. We can envision weighted object sets in two (generally isomorphic) ways: either as sets of <object, weight> pairs; or as structures $\langle S, \omega \rangle$, where S is a classical set of objects and ω a *weighting function* that maps each object in S to a non-zero weight. The first picture reflects most common implementations of posting lists and result sets; the second is more amenable to formal manipulation. We will move back and forth between them at will.

Most of the structure of a weighted object set derives from the underlying set S. As in a classical set, the objects in a weighted object set are unique. Weighted object sets take part in the classical set operations of creation, test for an element, intersection and union. In the case of intersection and union, the result is an underlying set that is the classical intersection or union of the component underlying sets, together with a weighting function derived by some *weight combining function* from the weighting functions of the component sets. Any ranked retrieval searcher implements a weighted object set union defined by some particular weight combining function.

The most marked enhancement in a weighted object set is that the weighting function implies a (pre-)order in the elements. Elements with a higher weight can be said to rank above elements with a lower weight, and it is often useful to explore a set in this order. Thus each weighted object set is equipped with an iterator that returns elements in non-increasing weight order. This iterator is not necessarily unique. When a set includes more than one object with the same weight, different iterators may return those elements in any order.

1.1. Posting Lists as Weighted Object Sets

If we explore a weighted object set in iteration order, we will see a sequence of non-decreasing weights. We can regard this sequence as a discrete function over the set, considered as an ordered collection. We will refer to this as the *weight attenuation function.* In actual posting lists, the attenuation function is always a stairstep function (Figure 1). Each stair tread corresponds to a particular combination of document characteristics. In systems using TF or TF-IDF (Term Frequency – Inverse Document Frequency) indexing, for instance, treads correspond to different term frequencies. Since there are only a finite number of document lengths in a given collection, and since for each length only a few term counts are possible, the function must necessarily be discontinuous. Actual posting lists, moreover, are samples from this space that reflect the distribution of text sizes in the collection and the distribution of term occurrences within text. Thus they have pronounced stairstep shapes, particularly in the central region corresponding to the most likely document sizes and term counts.

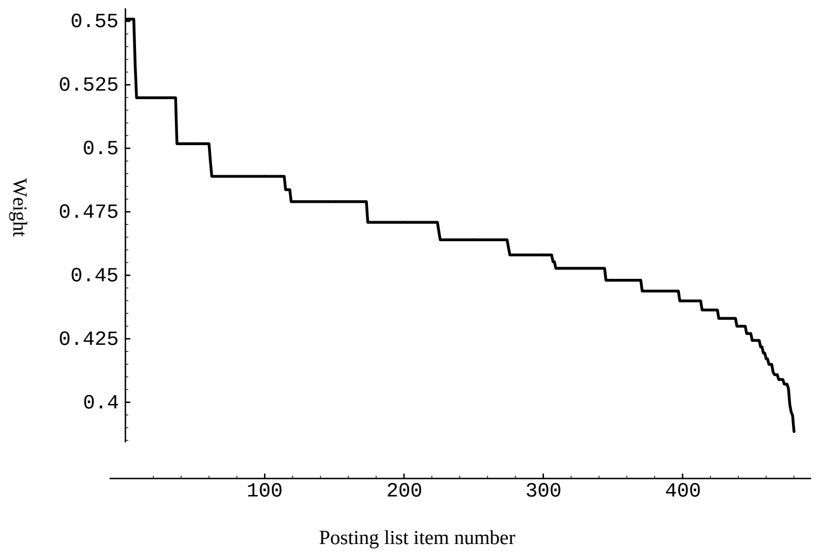

Figure 1: An actual posting list attenuation function. The posting list for “cat” among titles in the Virginia Tech library collection has 480 elements ranging in weight from 0.55 to 0.39. The weighting function is a variant of TF-IDF, so each stair tread corresponds to a particular combination of term count (almost always 1) and document length. The sigmoid shape is typical of a TF weighting function: the middle weights, corresponding to the most likely term counts in the most likely text sizes, have the longest treads, while the high and low weights, corresponding to very short and long texts, are represented by only a few objects.

The shape of a typical posting list curve depends on the weighting system used. Any monotonically non-increasing function can theoretically be achieved by manipulating the way weights are assigned to the objects during indexing. Indexing systems based on document characteristics, however, will produce curves that reflect the statistical character of the document collection. For instance, assume that document text sizes are normally distributed, and that any posting list is a random sample drawn from that distribution.

To facilitate statistical analysis of our model, we envision a text collection as if all the texts were strung together in some arbitrary order to form one long sequence of tokens. Each token is then considered a single event in a long event sequence, and the particular term used to index the token considered the outcome of the event. Basically, if there are T terms in the underlying language, we consider each token to be the result of a T-way choice, where the T different outcomes occur with widely disparate probabilities.

This is the model proposed by Shannon [SHANNON41]. It applies to growing document collections as well as it does to static collections. In Shannon's terminology, the document collection is a stream of tokens generated by an information source. The information source chooses terms from a language using criteria unknown to us, but following a definite probability distribution. We do not know the distribution *a priori,* but as the collection grows – as we get a larger and lager sample of the source's behavior – we can make better and better guesses of the generating distribution. In the case of text collections, no sample includes the full language; larger samples generally include terms never seen before, so that as a collection grows we gain information both about the underlying set of terms and the probability distribution within the set. It is a measure of the power of Shannon's model that this simple description covers so many situations so well.

If a text collection is a (relatively large) sample of an infinite stream of tokens, an individual document is a smaller sample from this same stream. The collection is partitioned into documents, each document being a contiguous sample of tokens (events) produced by the information source. Thus each document affords us a probability distribution that serves as a picture of the information source. The distance between this distribution and the distribution of the collection can be measured in a number of different ways [KAPUR92].

We assume that document lengths are normally distributed around some mean λ. This assumption fits both our intuitions and the VT Library collection data used in MARIAN (Fig. TEXT_LENGTHS). If we further assume that the occurrence of any particular term in Poisson – that later events with that term as outcome are not conditioned by earlier events – then we would expect the within-document distributions to be fairly close to the collection distribution. Recent research by Church [CHURCH95] has cast substantial doubt on the later assumption. In particular, Church hypothesizes that "interesting" terms – terms that reflect the content of the document – tend to deviate from Poisson production, while less significant terms continue to follow the Poisson model. While we find these results tantalizing, we will ignore them in this analysis, both because our dependence on a Poisson production model is low, and because preliminary checks [[DO THIS!]] indicate that non-Poisson terms are relatively rare.

These two assumptions allow us to predict the distribution of raw TF data – that is, pairs of $\langle ct_{Term}, ct_{Total} \rangle$ values – in the collection. In particular, if a term has a collection-wide frequency ϕ_t , it has an expected distance of ϕ_t^{-1} tokens between successive occurrences. The expected raw TF distribution reflects the interaction of variation in individual distances with variation in document lengths.

To take a specific example, choose a term t where $\phi_t^{-1} >> \lambda$, so that no document is expected to include more than one occurrence of t. This is the case for most terms in many collections. In a collection where λ=100, for instance, this applies to all terms that occur with a frequency substantially smaller than 0.01; that is, to all but a handful of terms in English text. When this condition obtains, we can ignore cases where $ct_{Term} > 1$, and the raw TF data degenerates to a simple [[DISCRETE-NORMAL]] distribution. Figure 2 shows the attenuation functions produced by several common weighting functions from this distribution of raw data.

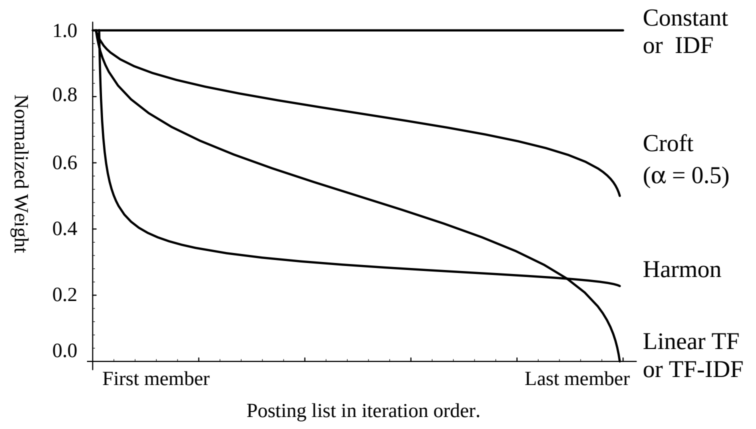

Figure 2: Attenuation functions for abstract posting lists generated by a normal distribution of text sizes and (where appropriate) term counts, using a variety of common indexing systems.

1.2. Ranked Retrieval as Weighted Object Set Union

A ranked retrieval system produces a weighted object set of all documents matching a given query, from those with the closest match at the top, down through any document that matches the query, however trivially. If the query is a flat or weighted collection of document components (as opposed to a statement in some calculus) then any document matching any component will be part of the result set. In particular, if we pretend that both queries and documents are simple pieces of text, any document that matches any word in the query will appear in the result set. Thus we can model a search engine of this type as a weighted object set union.

Search engines covered by this model include all those based on the *vector model* [SALT&McG83], some probabilistic search engines [CROFT83], information theoretic engines [vanRIJSBERGEN79], and many ad hoc implementations. These search engines draw on divergent models of the retrieval process, use different means of assigning posting weights, and use different weight combining functions. They have in common that the final weight of an item in the union is a function of the weights obtaining between the terms in the query and the documents in the collection. In particular, they all implement a weight combining function based on a sum. In the case of the vector model, this sum is a (normalized or unnormalized) inner product; in the probabilistic model, a sum of partial contributions to the estimated relevance; in the information theoretic, a calculation of the relative entropy or cross entropy of the document. The component weights may be manipulated before being summed, and the final weight is often a function of the sum, but the sum remains the central operation.

The intuition underlying summing weights is straightforward. All other things being equal, a document that matches more parts of the query should receive a higher rank than one that matches fewer. Of course, all other things are not equal: different query components have different differentiating ability, and a match on an infrequent term in the language, for instance, may count more than a match on a frequent term. This disparity is the motivating force behind the use of stopping rules. Low-weight members of low-weight components, it is argued, can be reasonably discarded with little effect on the final weights of the components at the head of the combined set.

If the weighting function of a union set is based on a sum, we expect the resulting weight attenuation function to be made up of segments. If two weighted object sets are combined, for instance, we expect elements from the set overlap to have noticeably more weight than elements that occur in only one component set. If the attenuation functions of the component sets are sufficiently steep, some high-weight single elements may end up higher in the ordering than some overlap elements, but as as tendency, the two segments will be distinct. This effect is clear in practice. In Figure 3, for instance, we see summative unions of two, four, and 27 posting lists for terms from veterinary medicine. When two sets are summed, the distinction between the overlap segment and the single-set segment is very clear. As more sets are added, the distinction between segments becomes less and less well defined. The curve for the 27 set combination is, in fact, almost completely smooth. This is only to be expected. With 2^{27} different possible subsets of posting lists with widely varying IDF values, almost any gradation is possible.

In order to model the behavior of a summative union, we must first develop the statistics that determine overlaps among the component sets. When the component sets are posting lists, this is governed by the *hypergeometric distribution* (see Appendix). In this section we will only be interested in the expected value (mean) for the distribution. When we examine stopping rules, we will also be interested in the *quantile:* the first value for which the cumulative probability density exceeds some value.

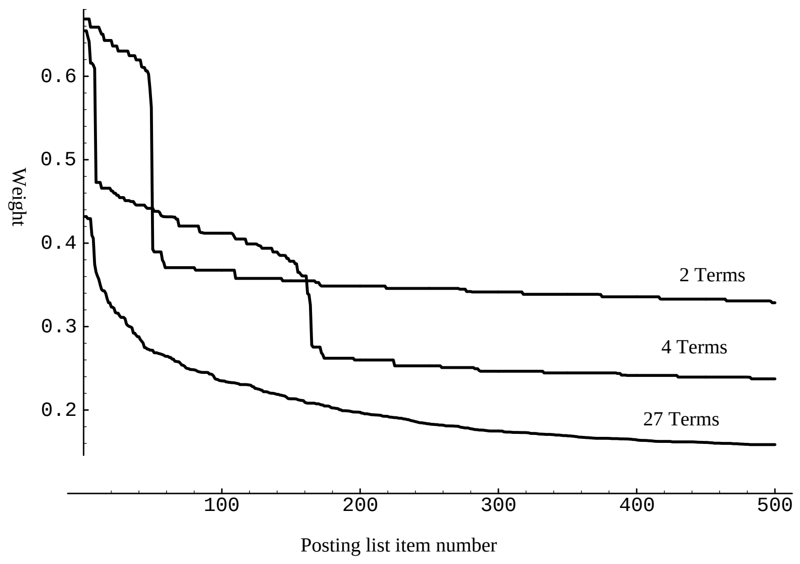

Figure 3: Initial sections of actual set attenuation functions for combined sets from the VT library title collection. In the first set, two terms ("cat" and "dog") produce a small high-weight area where the terms overlap, followed by a long tail. In the second, four terms ("cat", "dog", "anatomy" and "physiology") produce a tiny high-weight head, a moderate mid-weight section where two or three terms match, and then a long tail. A set of 27 terms from veterinary science produces a smoother curve with no apparent head.

When we take the union of two posting lists of size **n** and **m** in a universe of **N** items, we expect the size of their overlap to vary around the mean. In other words, we expect there to be

$$n + m - nm/N$$

unique items in the union, of which **nm/N** occur in both posting lists, **n-nm/N** occur in only the first, and **m-nm/N** occur in only the second (see Figure 4). With more component sets, we must consider all possible overlaps between pairs, then triples, and so on. To simplify matters, let us consider the case where all **k** component sets have a single size **n**. Let us further agree that each of the component sets have a constant weight attenuation function, and that the weight combining function of the union be a simple weighted sum, or average. That way the weight of any element in the union will be directly proportional to the number of component sets in which it occurs. The mean number of items found in all **k** sets will be

$$\text{overlap}_k = \frac{n^k}{N^{k-1}}$$

,

the mean number found in (k-1) sets will be

$$\text{overlap}_{k-1} = \binom{k}{k-1} \frac{n^{k-1}}{N^{k-2}} - \text{overlap}_k ,$$

and in general, the mean number found in i sets ($1 \le i \le k$) will be

$$\text{overlap}_i = \binom{k}{i} \frac{n^i}{N^{i-1}} - \text{overlap}_{i+1} .$$

Overlap size thus drops off geometrically as i increases. We expect no overlap at all to occur until $n^k \ge N$, no overlap of three sets to occur until $n^k \ge N^2$, and so forth. This is illustrated by the six-set case in Figure 4, and is consistent with the actual unions shown in Figure 3.

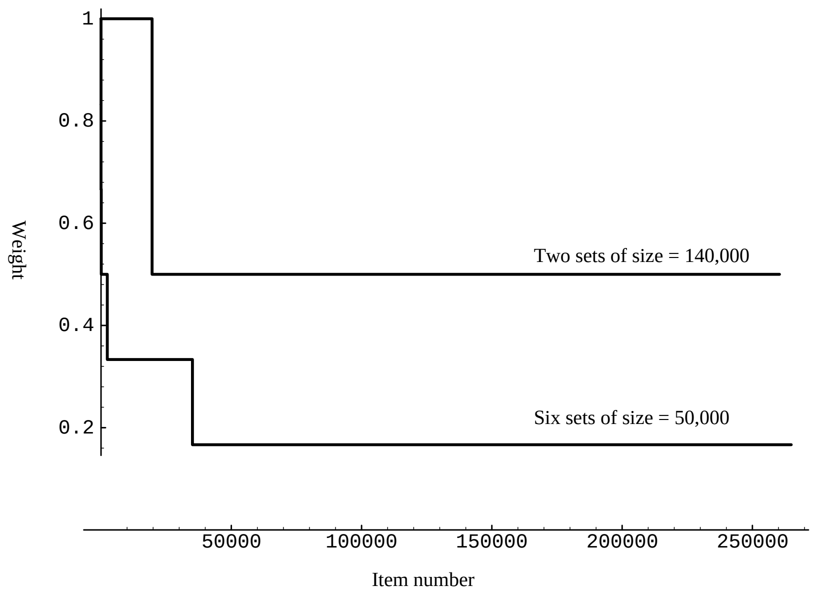

Figure 4: Expected shape of the attenuation functions for simple summative unions of two and six weighted object sets of equal size and constant weight (= 1). Collection size in both cases is 1,000,000, but the size of the component sets have been manipulated to produce coextensive curves. Note that overlaps of greater than three out of six sets have vanishing expected sizes. With these parameters, expected size of an overlap of all six sets is 0; of five, 2; and of four, 92. This is a fair number of documents for a user to examine, but miniscule compared to the total number of expected items in the union (264,908).

2. Opportunistic Search and Stopping Rules

Suppose that we are constructing the union of **k** weighted object sets using a simple or normalized sum as our combining function. We want to formulate our construction in such a way that we can stop before reaching the end of some or all of the component sets, if we find at some point that we have already established the initial segment of the union well enough to meet some heuristic criteria. This is what we mean by opportunistic search, and the criteria are our stopping rules.

During this process, there are two structures of interest: the collection of component sets and the set of items that we have already seen, which we will refer to as the partial set. In a real search engine, the first is likely to be a set of disk files; the second, some form of accumulator bank [HARMON92]. During the search process, items are run from the collection into the partial set. If an item is not already in the partial set, it is added; if it is, its weight is added to that of already accumulated. At any point in this process, the partial set is an approximation of final union set in two ways. Some items only occur in the unexplored portion of the component set collection, and are thus missing from the partial set. And any item in the partial set that also occurs in the unexplored collection has a weight lower than its ideal. As the exploration continues and the unexplored portion shrinks, this approximation gets better; when it gets good enough, we stop.

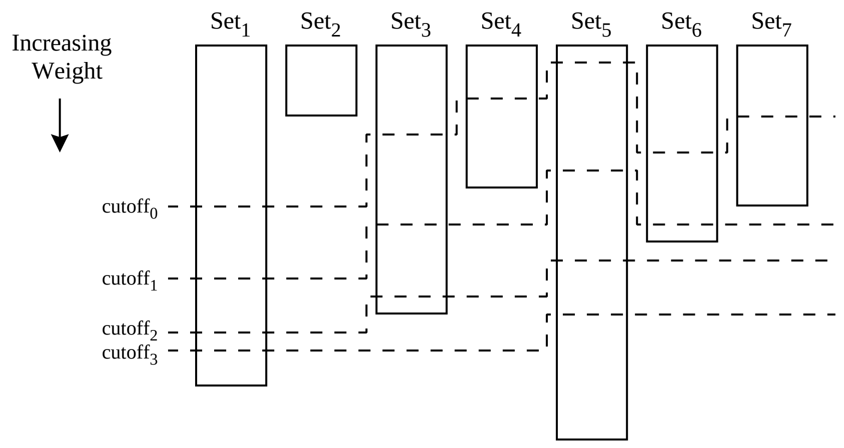

Figure 5: During opportunistic search, the MARIAN search engine explores a component set collection through regions of comparable weights. In each iteration, a cutoff weight is established and the sets are explored up to the frontier established by that weight.

Much attention has been placed in stopping rules research on the order of exploration of the component collection. Early systems [SM&vRIJS81, BUCK&LEW85] explored each list completely before going on to the next. Wong and Lee refined this by exploring each list a page at a time, always choosing the page in the unexplored portion with the highest initial posting weight as the next to run into the partial set. The MARIAN opportunistic search engine takes this process to its logical conclusion and explores the lists in a fully interleaved manner. (Paging is handled at a lower implementation level.) At each stage in the search, the engine draws a frontier through the entire component collection, all members beyond which have a weight below some cut-off value (Figure 5). The first cut-off is set quite low so that a good initial approximation can be made. Subsequent cut-offs are derived as a side-effect of applying the stopping rules. This order is easy to deal with in the abstract, so we will use it in our discussion.

When a new frontier has been reached and all the items encountered fed into the partial set, the top portion of the partial set is considered. There are two questions we can ask of this portion:

- Is the composition likely to change?
- Is the order likely to change?

The composition can change if there is an item in the unexplored portion of the set whose total accrued weight would be greater than the lowest weight in the initial portion, or if there is an item in the remainder of the partial set that would be promoted into the top portion by a “hit” from the unexplored component collection. The order can change if a “hit” occurs within the initial portion. In both cases, we can define “likely” to mean “within some range of probabilities.” A stopping rule makes a guess at how likely one of these events are to occur, and terminates the search if the probability is below some acceptable level.

3. Analysis: When Can We Stop?

In this section we evaluate the stopping rules defined above by examining the probabilities that exploring the rest of the way through a component collection will affect the top portion of a partial set. This can be broken into three questions: what is the probability that one or more elements discovered in the exploration will change the order of elements in the top portion; what is the probability that a combination of weights from the unexplored collection will cause a previously unseen object to invade the top portion; and what is the probability that one or more elements discovered in the exploration will promote an element from the lower portion of the partial set to the upper?

The first question can be re-cast simply as a question of sampling. Consider the top **t** elements of the partial set to be a sample (of size **t**) drawn without replacement from the set of all **N** documents. The unexplored portion of the component set collection is a sample (of size **c**) drawn from the same collection *with* replacement. We are interested in the probability that these two samples have an element in common.* For the moment, we will ignore the internal structure of the unexplored collection, and consider only the number of unique elements in it. We can use the hypergeometric distribution $h_{<N, t, c>}$ to calculate the chance that one hits the top portion.

Say we want to know whether the top 20 elements of a partial union set are stable at the .01 level; that is, whether there is a 99% chance that they will be unaffected by exploring the remainder of the component collection. Let us assume a universe of 1,000,000 objects and ask how small the unexplored collection needs to be to obtain this level of stability. We want to find a **c** for which

$$h_{<1000000, 20, c>}(0) \geq .99$$

Using the special case for x=0 (see Appendix), we find that the largest **c** for which the inequality holds is 502.

Remember that we are dealing here with posting lists of thousands, even tens of thousands of elements, and posting list collections of hundreds of thousands of elements. It is these numbers that drive us to use stopping rule methods. Once we have reduced such a collection to the point where only 500 elements remain, we are virtually done, and might as well load in the last few. The amount of computation time that has been spent in calculating probabilities and maintaining the extra structure needed for opportunistic search is already more than would be required to run 500 elements into an accumulator bank. Thus we conclude that to satisfy this criterion of stability we are best off doing an exhaustive exploration of the component set collection.

***** To be completely precise, we are interested in the probability that these two samples have an element in common where the total weight of the element in the unexplored collection is greater than the difference in weight between the element in the partial set and the element immediately above it: the probability that we will get a hit that displaces the item hit. In point of fact, however, the weight differences between neighboring items in a union set tend to be very small compared to item weights in the component sets (compare Figure 3). When the item is on a stair tread, there is no difference between neighboring elements. So we are not far off in asserting that *any* hit in the initial segment will displace the element hit.

Fortunately, there is ample justification for relaxing this criterion. Consider the case of a simple text search, where a user is attempting to match a vector of terms across a text collection. No known ranking function does a very good job of ordering the results of such a search, although many correlate well over a large sample with users' judgements (for an overview of related research, see [SARACEVIC91]). So we may well believe that we can give up precise ranking in the set *if we can be certain that the best documents are in the top portion of the set.* In other words, if we have established the content of the top portion correctly, we can afford to be relaxed about the order.

It is one of the features of the hypergeometric distribution that even for large N only small sets are at all likely to be disjoint. Figure 6, for instance, shows the probability that two samples drawn from a universe of 10,000 events will be disjoint. As can be seen, unless at least one sample is very small (less than a few hundred elements) there is virtually no chance that the pair will be disjoint. This "handball court" shape is less pronounced with smaller total universes, but the effect is the same: unless the samples are extremely small, the chances are that they will have some overlap.

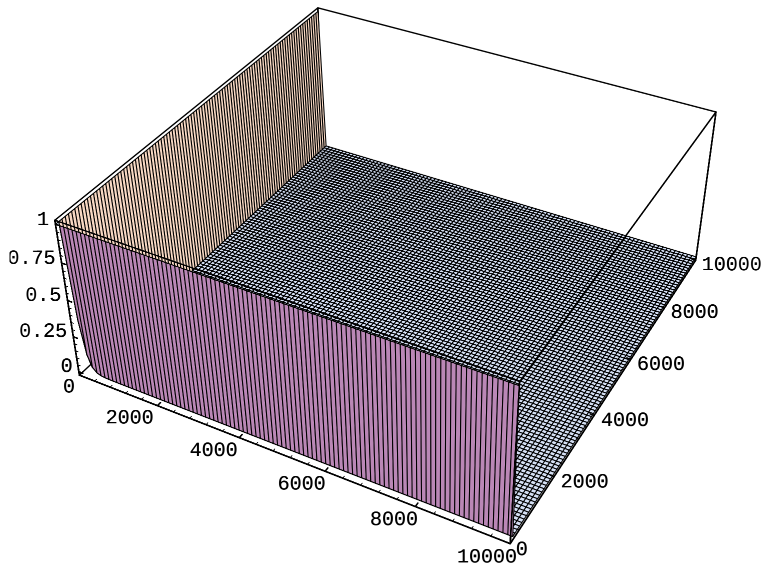

Figure 6: The "handball court" effect for the probability of two sets being disjoint. Shown here are set sizes from 1 to N (10,000) on the two horizontal axes, and the probability that the two lists are disjoint on the vertical axis.

The second of our tests asks whether any elements of the unexplored portion of the component set collection are likely to belong in the top portion of the combined set. We presume that this can only happen for elements that occur in more than one among component set. After all, we have already transferred a large number of elements of higher weight than those remaining. It is easy to test whether any combination(s) of expected weights from unexhausted component sets would be greater than the least weight in the head of the partial set. If so, it remains to calculate the expected size of those overlaps. This is a special case of the calculation for the third test, and any exploration that passes the third will pass the second.

The third test depends on the probability of a promotion from the lower portion of the partial set into the top portion. We want to know whether the dividing line between these two sections of the union is **stable** in the following sense:

> **Definition:** A point in the partial set is *stable* if none of the items below that point are likely to be promoted above it during exploration of the remainder of the component set collection. In particular, a point is *stable at a given level* if the expected probability of such a promoting hit is less than that level.

Note that a stable point is only that: a point. A point a few elements to one side or the other may not be stable, particularly if the stable point is on a stairstep riser. This parallels our discussion of the first test above. Just because a section of the combined list is stable – unlikely to receive any new elements during the remainder of the exploration – does not mean that it is static – that elements within the section are not likely to be reordered by further exploration.

We break the third test into a sequence of tests. First, we consider single hits from the remainder of the component set collection, and ask how many of these are likely to move an element from the tail of the partial set into the head. This is a sampling question, but not simply a question of the probability of overlap between the unexplored collection and the tail. It may well be the case that most of the weights in both the unexplored portion and the combined tail are very low, much smaller than our candidate point of stability. We are concerned only with a subspace of all the possible hits and misses: with only those hits whose combined weight is greater than the candidate point.

Next, we estimate how many objects in the unexplored collection occur in two lists, calculate an expected weight for them, and count potential promoting hits for those. Objects that occur in two component sets will have higher weight than those that only occur in one, so there will be more items in the tail that may be promoted. On the other hand, there will be fewer objects that occur twice in the unexplored collection. We continue by estimating the number of objects in three component sets, and so forth.

Practically speaking, we generally only need to ask the question for items from single component sets and from pairs of sets. As mentioned in Section 2, the expected size of overlap segments decrease geometrically as the number of sets involved goes up. Suppose that P_{cutoff}, the certainty that we want to achieve, is .999. (In real applications, we would never be this precise. The error in our indexing functions is already greater than .001; a more realistic level is .9 or .95.) If the component sets are posting lists for terms in ordinary language, the largest we can expect them to get is in the range 0.01N–0.1N. Even at the largest values, it is barely possible that we should need to consider triples of sets; certainly we need consider nothing less common. MARIAN opportunistic search uses two heuristics:

- Assume that all the items remaining in the unexplored portion are unique items. Count the number of potential promotions in the tail of the partial set. What are the chances that these two intersect?
- Calculate the expected number of items in the unexplored collection that occur in two lists. Assume that any of these will promote any item in the tail. What are the chances of a hit?

If both these probabilities are sufficiently low, the search engine can stop the exploration. These are cost-effective approximations to the more rigorous sequence of calculations in this paper.

The first calculation in our sequence is of the probability that the expected weight of a single unseen item will promote any item in the partial set's tail. In particular, if there are **N** possible items in the collection as a whole, $\mathbf{c_1}$ items that occur only once in the unexplored collection and **d** items in the tail with weight greater than the difference between the expected weight and the least weight in the top portion, we want to know if

$$h_{<N,\, c_1,\, d>}(0) > P_{cutoff}.$$

As before, we are only interested in opportunistic search when **c** is quite large. $\mathbf{c_1}$ is smaller than **c**, but not appreciably. This means that we can only stop when **d** is quite small (see Figure 7). In other words, the only places on the partial set attenuation curve where this condition is met are those where the average slope is very large: where one can achieve a large decrease in weight over very few elements. When we are combining only a few sets, the union attenuation function is still primarily a stairstep function (see Figure 3). In this case, we are only likely to be able to stop when our candidate stable point is close to the trailing edge of a stair tread. Such stable points do exist, but they are unlikely to exist at predetermined points. In particular, if we are trying to establish stability of the first element, or the tenth or twentieth, we are probably out of luck.

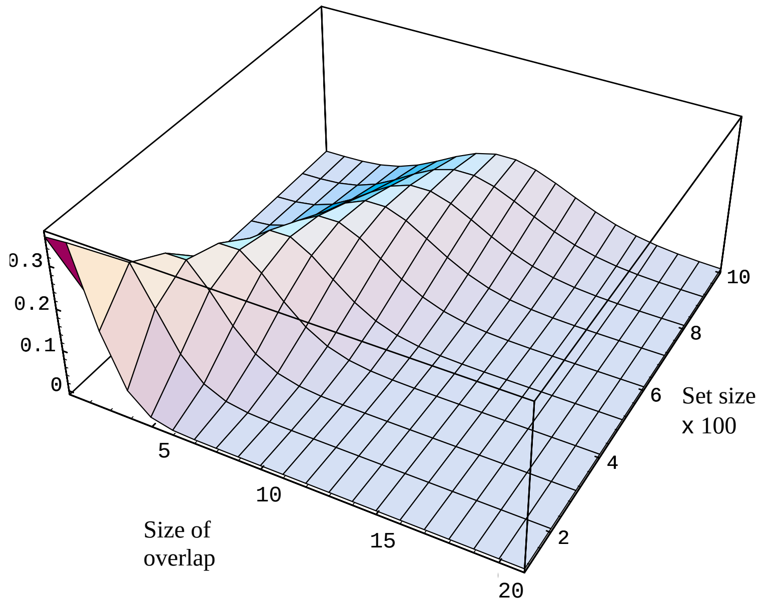

Figure 7: Probability density surface for overlaps between a 1000-element set and sets of sizes from 100 to 1000 in a universe of 100,000 items. For the case of $h_{<100000,\ 1000,\ 100>}$, overlaps of 0 and 1 are most likely. By $h_{<100000,\ 1000,\ 1000>}$, the mean overlap is 10 items. Even in this case, though, the probability of an overlap of size 20 or greater is still less than .002.

When many sets are combined in an opportunistic union, the situation is more encouraging. In the 27-set union from Figure 3, for instance, there is a large initial region where the average slope is uniformly high. Any candidate point in this region has the required characteristic that a single hit from one of the 27 lists will promote very few elements below it. In the later region of the attenuation curve, this is no longer true, but this region is unlikely to be explored by any human user (see Figure 8).

A similar analysis applies for objects that occur in several of the component sets. If an item occurs in two or more component sets, then we need to know if the sum of the expected weights for those sets is likely to cause a promotion. Since the total hit weight is higher, we will need to include a larger portion of the tail in our one sample; however, the expected number of items in common between component sets is likely to be much lower. The hypergeometric distribution again gives us a maximum number of acceptable promotions. Together, the numbers define a second line that the curve must also fall below (see Figure 8). This condition may be either more stringent or less than the first. In the normal case for ranked retrieval, where the number of component sets is small and their sizes less than 0.1N, the expected overlap will generally be an order of magnitude smaller than the expected number of single hits. The change of weight, on the other hand, is comparable. For a simple summative union, it is in fact twice the expected weight for a single unseen item. So any situation satisfying the test for single hits will pass the test for multiple hits.

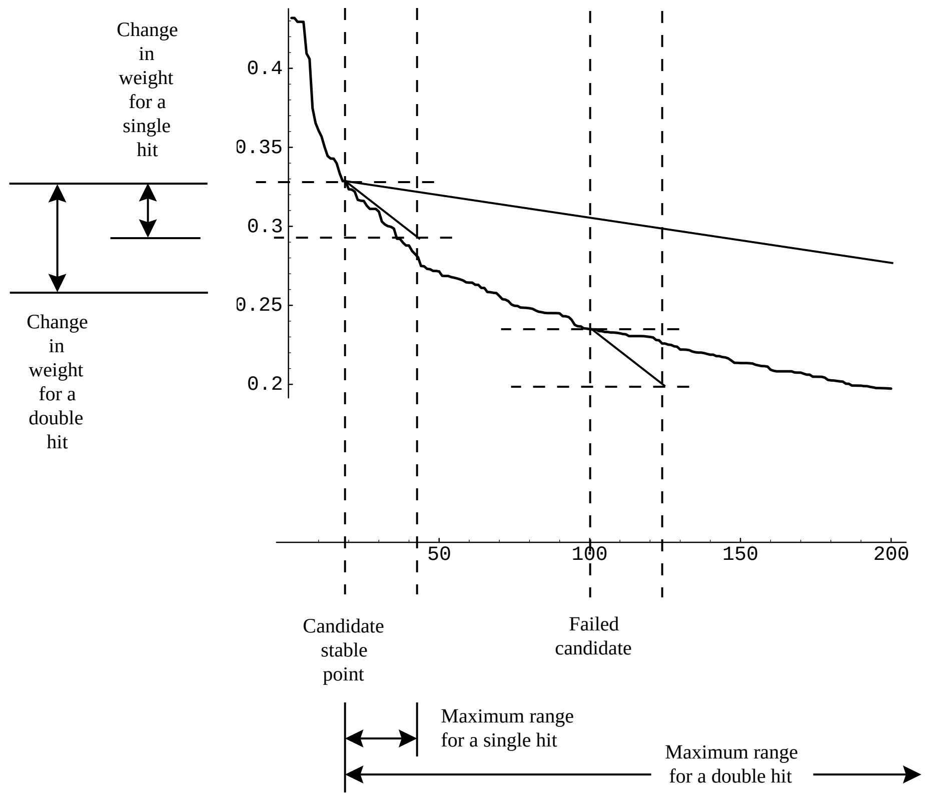

Figure 8: In this magnification of the initial segment of the 27-set union from Figure 3, two candidate stable points are shown. The candidate at item #20 meets the abstract stopping rules for stability, as fewer points below it in the union iteration would be promoted by a single hit than the maximum number. The point at item #100 fails the same test. Also shown is the test for pairs of hits. Note that while the change in weight for this test is only twice that for the single hit test, the maximum acceptable range is many times greater.

This situation only reverses in two cases. One case – when component set sizes approach the collection size – does not occur in text retrieval. The other case is when large numbers of component sets remain unexplored. In this case, the test for multiple hits becomes more stringent than that for single hits, simply because the number of multiple hits is on the same order as the number of single hits.

4. Discussion

The preceding analysis presumes statistical independence of the component weighted object sets. This is a common assumption – and in fact the only assumption that allows a tractable analysis – but in point of fact it is completely mistaken. Words in English have complex networks of interdependency [DEER&al90]. Words in a query are if anything less independent. We might say that users construct queries in the fervent hope that their search terms are *not* independent; that they are likely to co-occur in documents of interest.

Suppose we are searching a moderate-size research library collection (the Virginia Tech collection of 1994, with a total of 960,000 volumes) using the query **Author: Asimov; Title: foundation**. In the Virginia Tech collection there are two authors named Asimov: Isaac, with a total of 137 books to his credit, and Janet with two. The word "foundation" in the title retrieves 3754 books, including nine by Asimov. Let us pretend that these two keys are statistically independent and calculate the odds of any overlap at all.

As developed above, the odds of an overlap of x works are

$$h_{<900000,\, 139,\, 3574>}(x) \quad 0 \le x \le 139$$

With these parameters, the odds are 0.56 that the two sets have no intersection at all. Above x=3 the odds become miniscule, and the probability of an overlap of nine works calculates to 9.00×10^{-9}. Such an event thus satisfies the most stringent tests for statistical significance – not surprisingly, since there is in fact a significant correlation in the real world between Isaac Asimov and the word "Foundation." It is that correlation that informed the query, and that undermines any statistical rules for curtailing exploration.

This does not mean that stopping rules are useless. We noted above that the rules favor situations with large numbers of component sets. An opportunistic search on the 27-term query in Figure 3, for example, stops relatively early in the process. In applications such as clustering and relevance feedback, where full-size documents are used as queries, opportunistic search can make a difference. Typical queries authored by users, on the other hand, tend to be short and to include fewer high-frequency terms than ordinary unconstrained language. These queries invoke precisely the statistical patterns that mitigate against stable points in the upper portion of the union. So it is no surprise that when user-supplied queries are run against large document collections, stopping rules fail to stop.

5. Acknowledgements

This paper and the MARIAN system both owe much to Dr. E.A. Fox. Ed conceived of MARIAN as a production system, and has championed it though a long and complex development process. He also is responsible for introducing me to the method of stopping rules, in particular to the research of Wong and Lee, and for recommending that we include it in MARIAN. He has been extremely encouraging throughout this research; I hope that he does not find the results herein too *dis*couraging.

Both the Virginia Tech Computing Center and the Tech Library have supported MARIAN; the Library by selflessly sharing their data, and the Computing Center by funding the development process. Too many individuals in the Center and in the Department of Computer Science have contributed to MARIAN to mention here. I will make a single exception and thank Dr. Ben Cline, without whom the effort would never have gotten off the ground.

Mike Keenan pointed out to me that the hypergeometric distribution was “the one I was looking for.” My thanks go to him and to the rest of the Dead Logicians Society for sitting through earlier presentations of this research and making valuable suggestions.

Bibliography

[BUCK&LEW85] Buckley, Chris and Alan F. Lewit, “Optimization of inverted vector searches.” *Eighth Int’l. Conf. on Research & Development in Information Retrieval (Montreal,1985).* 97-110.

[CROFT83] Croft, W. Bruce, “Experiments with representation in a document retrieval system.” *Information Technology: Research & Development* **2**:1 (1983) 1-21.

[DEER&al90] Deerwester, Scott, et al., “Indexing by latent semantic analysis.” *JASIS* **41**:6 (1990) 391-407.

[FOXetal93] Fox, E.A. et al., “Development of a modern OPAC: from REVTOLC to MARIAN.” *Proc. SIGIR ‘93: 16th Int’l. Conf. on Research & Development in Information Retrieval (Pittsburgh, PA, 1993).* 248-259.

[HARMON92] Harmon, Donna, “Ranking algorithms.” In Frakes, W.B. and R. Baeza-Yates, Eds, *Information*

Retrieval: Data Structures and Algorithms. New Jersey: Prentice-Hall, 1992. 363-392.

[HARM&CAND90] Harmon, Donna and Gerald Candela, "Retrieving records from a gigabyte of text on a minicomputer using statistical ranking." *JASIS* **41**:8 (1990) 581-589.

[LEE&WONG91] Lee, Dik Lun and Wai Yee Peter Wong, "Partial document ranking by heuristic methods." *Proc. Int'l. Conf. on Computing and Information (Ottawa, May 27-29, 1991)* 231-239.

[LUCARELLA88] Lucarella, Dario, "A Document retrieval system based on nearest neighbor searching." *Journal of Information Science* **14** (1988) 25-33.

[RASMUSSEN92] Rasmussen, Edie, "Clustering algorithms." In Frakes, W.B. and R. Baeza-Yates, Eds, *Information Retrieval: Data Structures and Algorithms.* New Jersey: Prentice-Hall, 1992. 419-442.

[SALT&BUCK88] Salton, Gerard and Chris Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management* **24**:5 (1988) 513-523.

[SALT&McG83] Salton, Gerard and M. McGill, *Introduction to Modern Information Retrieval.* New York: McGraw-Hill, 1983.

[SARACEVIC91] Saracevic, Tefko, "Individual differences in organizing, searching and retrieving information." *Proc. ASIS '91 (Washington, DC, Oct 27-31, 1991).* 82-86.

[SM&vRIJS81] Smeaton, A.F. and C.J. van Rijsbergen, "The Nearest neighbor problem in information retrieval: an algorithm using upperbounds." *Proc. Int'l. Conf. on Information Storage and Retrieval (SIGIR) (Oakland, CA, 1981)* ACM. 83-87.

[vanRIJSBERGEN79] van Rijsbergen, C.J. *Information Retrieval* 2nd Ed, London: Butterworths, 1979. See esp. pp. 130-139.

[WONG&LEE91] Wong, Wai Yee Peter and Dik Lun Lee, "Implementations of partial document ranking using inverted files." Unpublished report, August 1991.

Appendix: The Hypergeometric Distribution.

The *hypergeometric distribution* is a discrete probability distribution constructed by sampling a universe of N items without replacement. It is usually explained as the distribution of probabilities among the number of successes in a sample of n items, where m of the N items in the universe count as successes. For our analysis, however, it is more intuitive to use an equivalent formulation based on the size of the overlap between two subsets. In this formulation, the hypergeometric distribution $\mathbf{h}_{<\mathbf{N},\, \mathbf{n},\, \mathbf{m}>}$ is the distribution of probabilities among all possible sizes of the overlap of two sets of sizes **n** and **m** in a universe of **N** items. The distribution is defined over all possible sizes of the intersection, from 0 to the the lesser of **n** and **m**. We will use the notation $h_{<N,\, n,\, m>}(x)$, $0 \le x \le \min(n, m)$, to denote the probability of the two subsets sharing exactly **x** items. That probability is given by the equation:

$$h_{<N,\, n,\, m>}(x) = \frac{\binom{m}{x}\binom{N-m}{n-x}}{\binom{N}{n}}$$

$$= \frac{n! \cdot m! \cdot (N-n)! \cdot (N-m)!}{x! \cdot (n-x)! \cdot (m-x)! \cdot N! \cdot (N-n-m+x)!}$$

We are often interested in the probability that two subsets are disjoint; that they have no overlap. This value is given by

$$h_{<N,\, n,\, m>}(0) = \frac{\binom{N-m}{n}}{\binom{N}{n}}$$

$$= \frac{(N-n)! \cdot (N-m)!}{N! \cdot (N-n-m+x)!}$$

Computationally, the last formula looks like m factors (or n factors; we choose the smaller number for obvious reasons) of a steadily decreasing fraction:

$$= \frac{(N-n) \cdot (N-1) \cdot \; ... \; \cdot (N-n-m+1)}{N \cdot (n-1) \cdot \; ... \; \cdot (N-m+1)}$$

$$\cong \left(\frac{(N-n)}{N} \right)^m$$

Where this last, of course, is the value for the binomial distribution obtained by taking two samples *with replacement* from a universe of N items. The larger N is with respect to n and m, the better the binomial distribution approximates the hypergeometric.

The binomial and hypergeometric distributions are intimately related. They have the same mean:

$$\mu = \frac{n \cdot m}{N} \quad .$$

The hypergeometric distribution, however, forms a sharper peak than the binomial distribution, growing sharper still as the sample sizes grow. Where the variance of the binomial distribution is given by $\sigma^2 = n{\cdot}(m/N){\cdot}(1 - m/N)$, the variance of the hypergeometric distribution is

$$\sigma^2 = \frac{n \cdot m}{N-1} \left(1 - \frac{n}{N} \right) \left(1 - \frac{m}{N} \right) \quad .$$