SMASER

Non-Boolean Searching in TRIP

White Paper

Abstract:
This whitepaper introduces the “best match” search functionality, describes its basic operation and the fundamental principles on which TRIP’s non-Boolean algorithms are based and shows examples of the function in use.

End User License Agreement

Table of Contents

TRIP and non-Boolean Search

TRIP has long offered a fully featured set of Boolean search capabilities, both generic and linguistic, but with the exception of the "fuzzy logic" introduced with the FUZZ command, has not offered a "best match" search capability.

TRIP has also a way of performing "best match" searches using a CCL function. This whitepaper introduces this search function, describes its basic operation and describes the fundamental principles on which TRIP's non-Boolean algorithms are based, and shows examples of the function in use.

Best match searching

Whereas Boolean searching focuses on matching a request with a set of provably valid responses (using basic set theory), best match searching focuses on matching a request with a set of responses that reflect the underlying purpose or intent of the request, rather than any provably correct "answer". From this set of possible matches, the algorithm then provides a weighting, or judgment, as to how relevant each matched document is when compared to the information need expressed in the query.

There are many different techniques used in common practice that aim to provide this level of inference between the query and the corpus of available information. Underlying almost all of these techniques, however, tends to be one of two prevalent algorithmic families: geometric or probabilistic measures.

Although TRIP's non-Boolean framework is designed in such a way as to allow for any type of algorithm to be used on a pluggable basis, the first such algorithm implemented follows the geometric form of measuring result relevance.

Geometric search evaluation

As the longest-established mechanism for evaluating best match queries, geometric algorithms (also called the cosine measure) have shown themselves to be remarkably capable over time (although the division between geometric and probabilistic backers tends to be rather unscientific and emotional).

The basis of the geometric method is being able to represent the information need expressed within a query and the information content of a candidate search result as vectors within a high-dimensional unit hyper-sphere (or, more practically, a projection onto an arc of that sphere). Assuming this can be performed, the similarity of the query and candidate can be calculated using a simple Euclidean measure of separation.

Figure 1 shows a very simple example of such an evaluation.

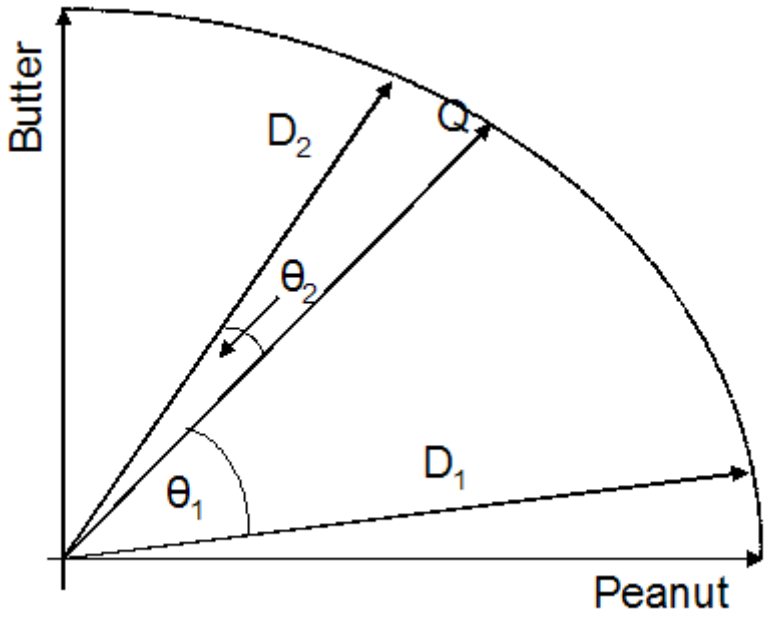


Figure 1

In this case, there are only two terms in the query ("Peanut butter"). For sake of simplicity, we have applied equal weight to each term in the query, resulting in a vector representing the query at 45° to each axis.

Again for simplicity, we are considering just two candidate documents (documents which contain either "peanut", "butter" or both).

For sake of illustration, document D_1 has been assigned a high "peanut" weight (implying a high degree of relationship between the document and the term or concept "peanut") and a low "butter" weight. This could happen, for example, if the document were about the health benefits of peanuts, and mentioned peanut butter only in passing. Document D_2, in contrast, has been assigned a high "butter" weight and a medium "peanut" weight.

When evaluating the similarity of the information content within documents D_1 and D_2 to the information need expressed by the query Q, we measure the angle between the vectors.

In this case, the angle θ_2 is much smaller than the angle θ_1 (in practice we actually seek to maximize the cosine of the angle as this is a self-normalizing function) and so we can state that document D_2 has a smaller divergence, or a greater similarity to the information need expressed in query Q than does D_1.

Obviously what separates one implementation of the cosine measure from another is how the implementation assigns or combines weights for terms or concepts in the query and in candidate documents.

Performing a best match query

As mentioned in the abstract, TRIP5 introduces a new CCL function to expose access to the best match framework:

```
Find ABOut(peanut butter)
Find ABOut(T=n [TO m])
Find ABOut(R=n [TO m])
Find ABOut(S=n [TO m])
```

As shown, there are many different ways of exercising the function. The simplest is to state an information need explicitly, i.e. to give the function the text of the query that you wish to match. This query text could be a simple phrase, a full sentence, an arbitrary fragment, or an entire page of text. Likewise, one or more terms from a Display list can be submitted as an information need, using the normal "T=" syntax.

An application wishing to offer a "find more like this one" function would make use of the "R=" syntax, whereby the content of the specified record is analyzed for information need.

Finally, in certain constrained circumstances it may be useful to find documents that match the information need expressed by all records in one or more search sets, using the normal "S=" syntax. Note that using this capability on large search sets will result in extremely poor performance, obviously.

Setting up a database to support best match query

In either of the latter two cases, the algorithm needs to know what attributes of the records should be analyzed when attempting to construct the information need being expressed by those records.

Equally, when indexing documents that will be tested for best match during queries of any of the forms shown above, the indexing engine needs to have this same information available.

The manager of such a database, therefore, must establish a new indexing flag on any field that is to be included in a non-Boolean calculation (either at indexing or query time). This is performed via TRIPmgr in one of two different ways:

Single field assignment is achieved by using the context menu on that field and setting the appropriate flag in the field's property page, as shown in Figure 2, overleaf.

Multiple field assignment is achieved by selecting all fields to be modified, and using the appropriate option from the context menu, as shown in Figure 3, overleaf.

Note that these options are only valid and available for fields of type TExt or PHrase.

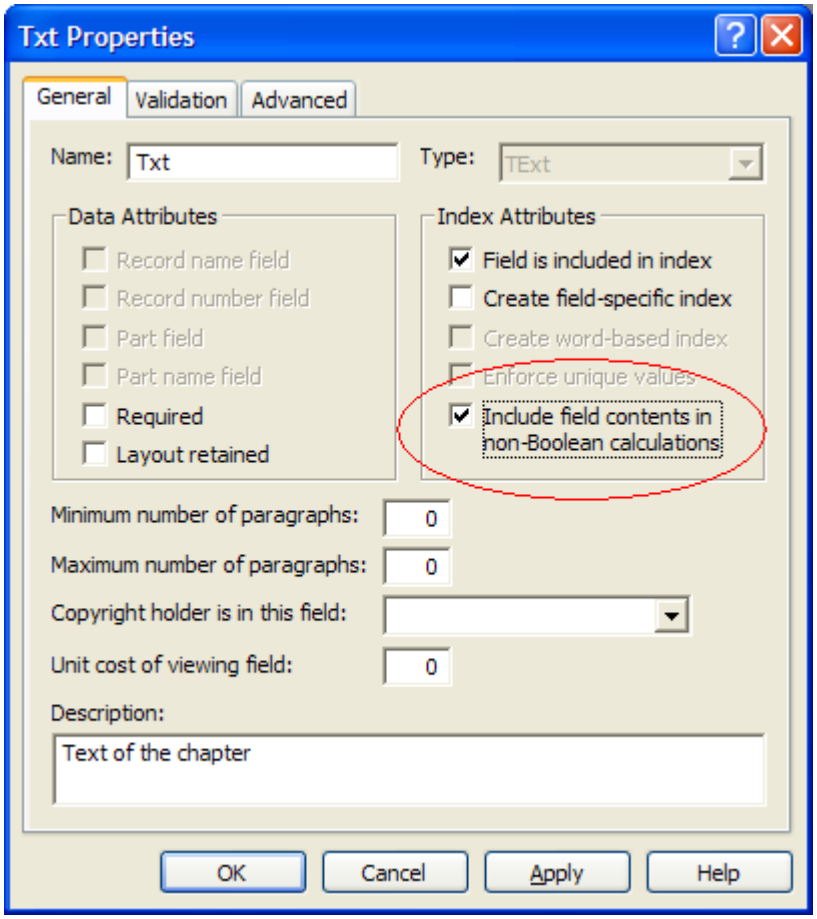


Figure 2

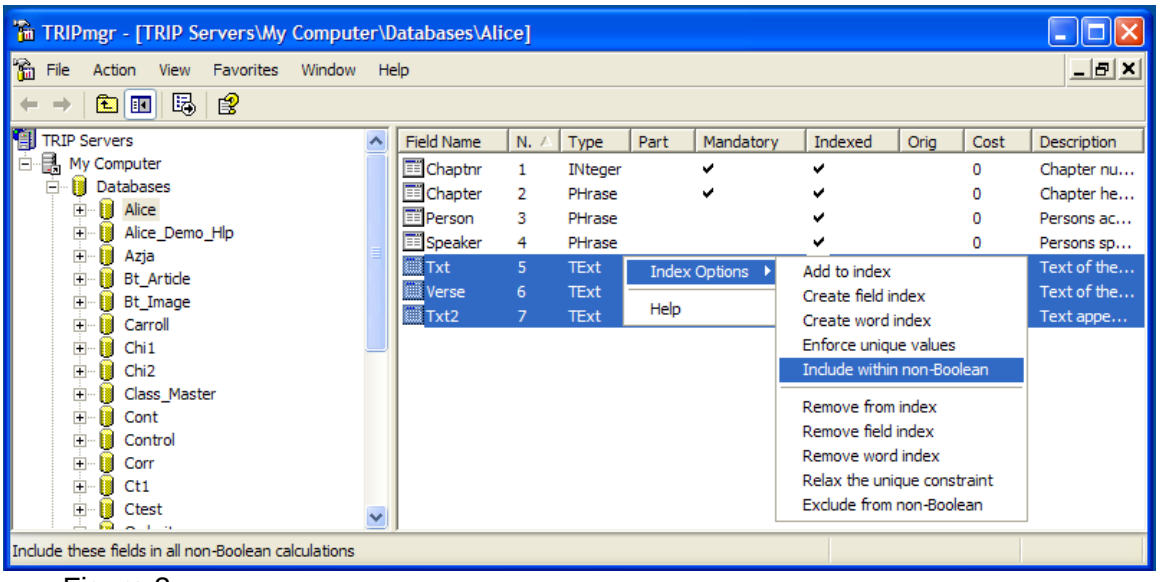
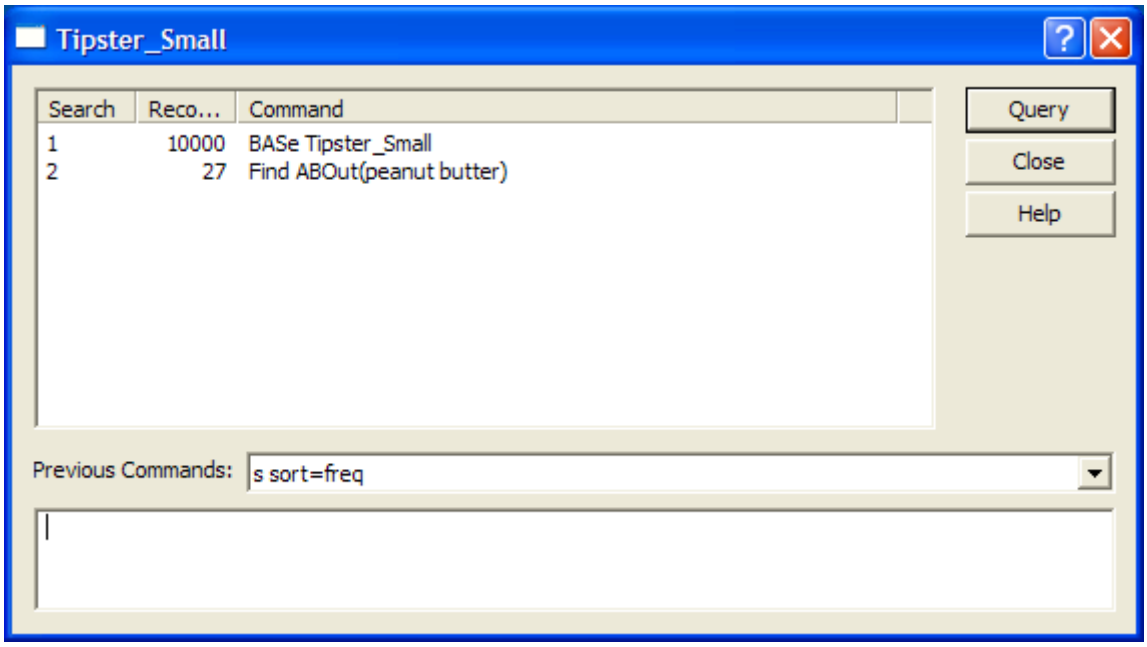


Figure 3

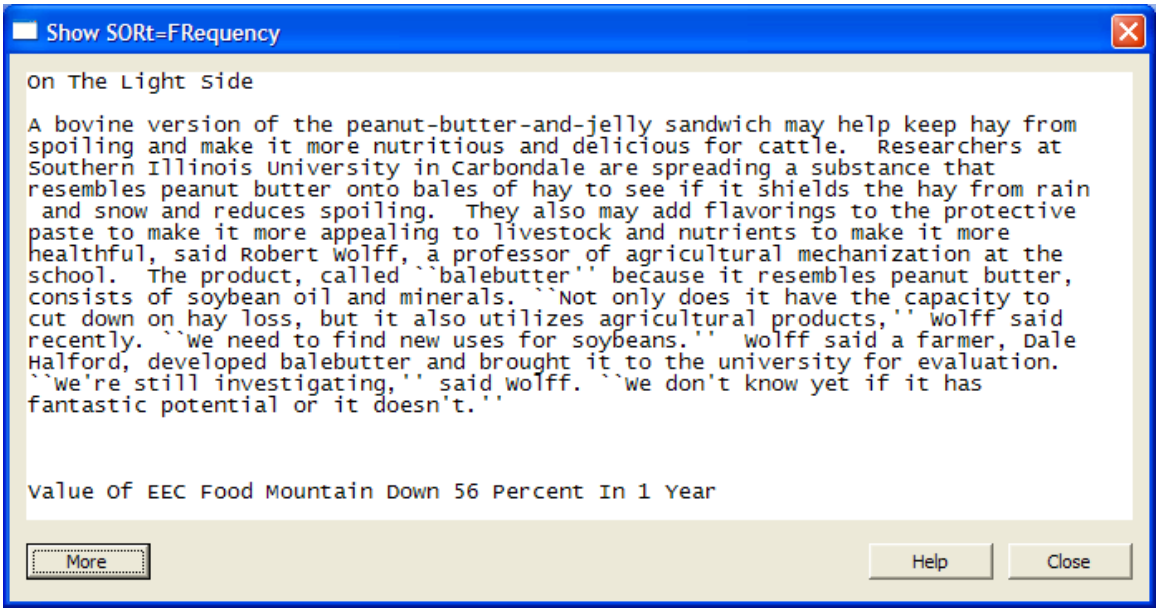
Once the flags have been set appropriately, databases that are to be searched using the new search function must be re-indexed in order for required information to be added.

Examples

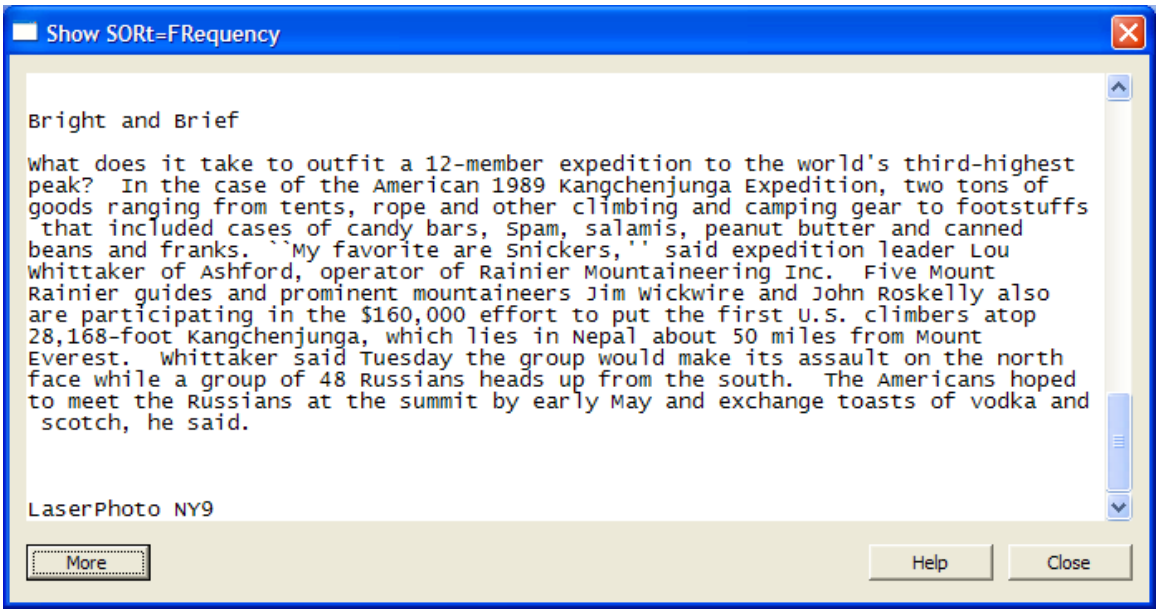
The following figure shows the new search function in action, performing the same basic query that we used in the example above.



Using the normal “Show SORt=FReq” mechanism of getting ranked output, results in the following document appearing first:

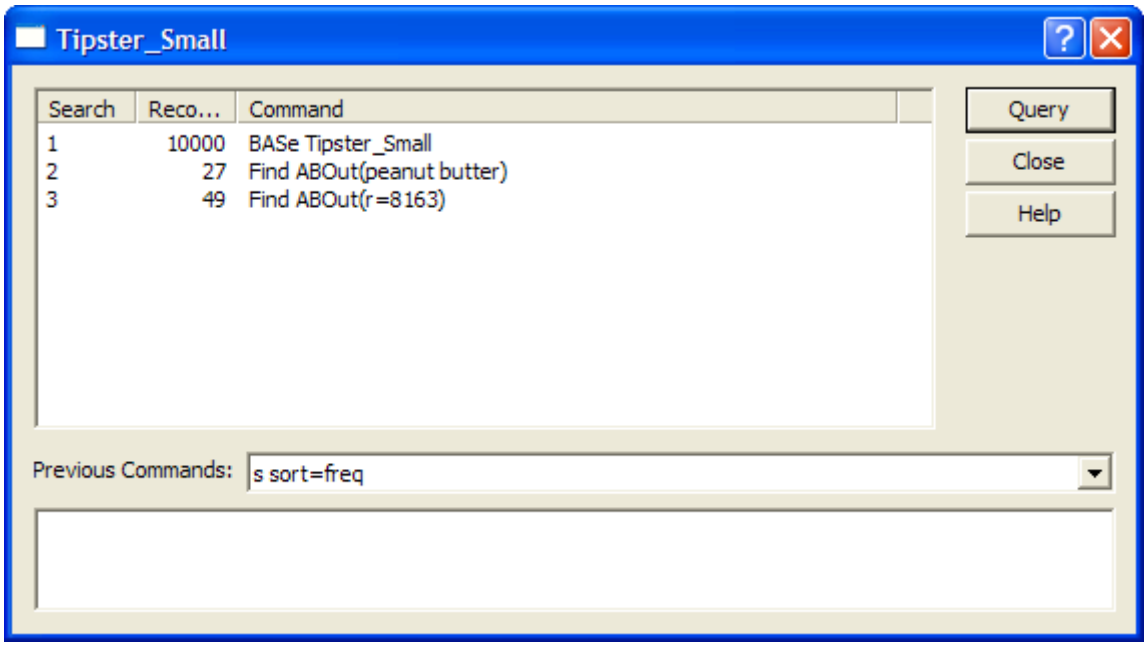


Further documents display more information about this topic, as expected, for example:



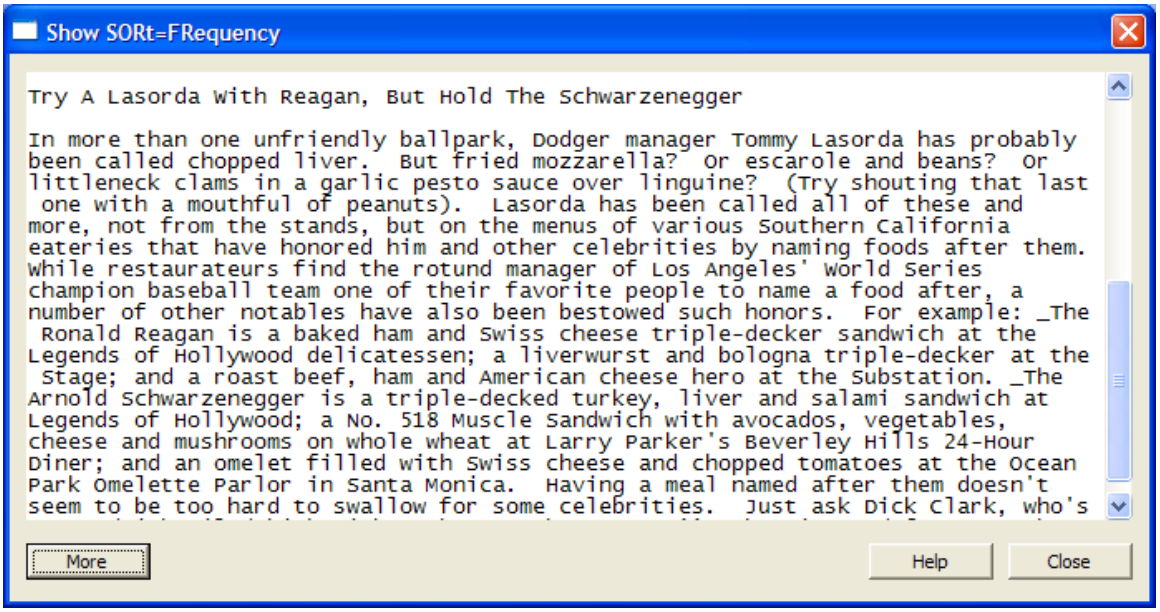
Whilst nicely ranked, this is nothing that a simple search for “peanut OR butter” couldn’t have done, of course.

Taking the example a step further, we submit a request to find more articles related to the information content within the first document shown:

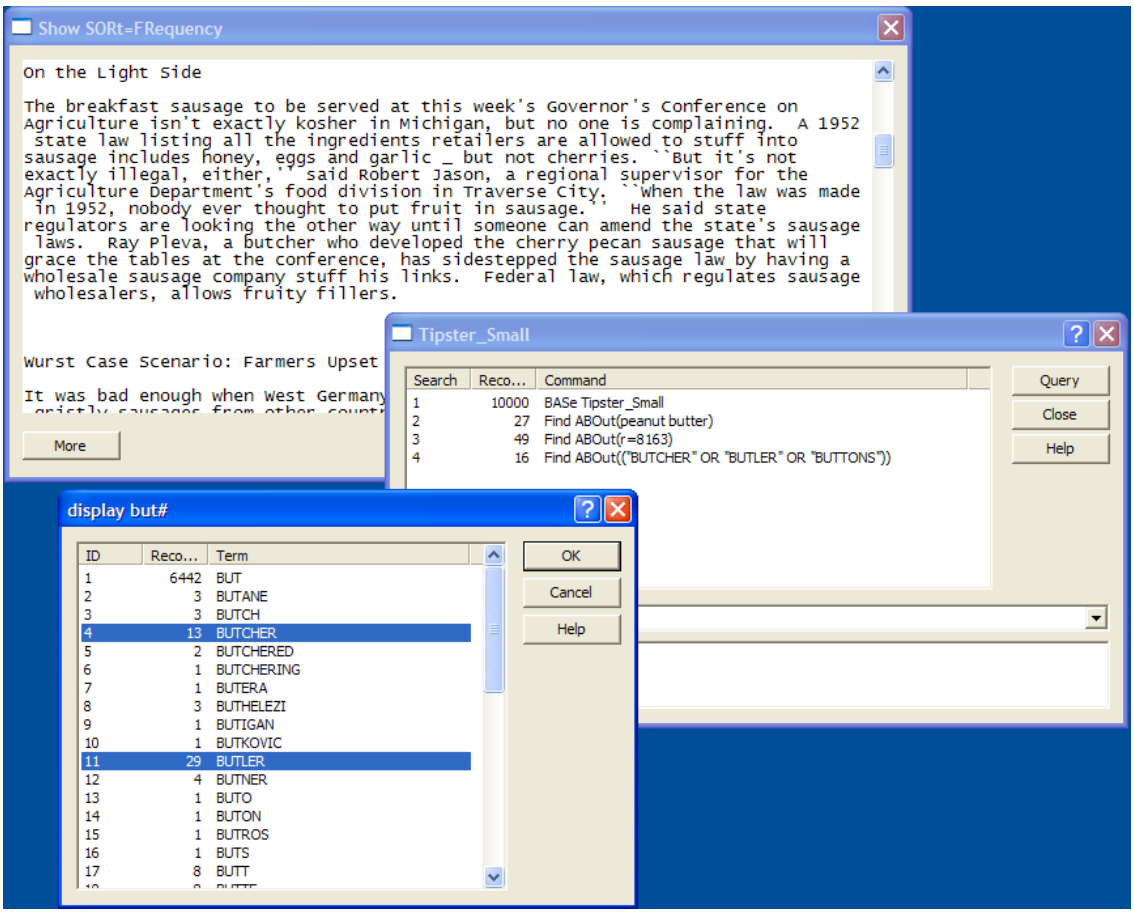


As should be expected, the first document shown is of course record #8163 (the most similar document should, after all, be itself).

However, the second document shown relates to other sandwich types being offered at a local ballpark:



This function can also be used to prepare a ranked list of documents that match terms from a Display window. For example:



Summary

TRIP offers the user a way of performing non-Boolean search in databases that have been prepared appropriately. The implementation of non-Boolean searching in TRIP uses a robust, proven algorithm family that should provide good, consistent results for a wide variety of contexts.