TREC-4 Experiments using DRIFT

Charles L. Viles and James C. French

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
{viles,french}@virginia.edu

Abstract

DRIFT is a prototype, vector space based, information retrieval system in development at the University of Virginia. The system is designed to do experiments in distributed, dynamic information retrieval. We describe our first experiments using DRIFT on larger test collections, specifically the Category B subset of the TREC corpus.

1 Introduction

The intelligent application of classic Information Retrieval (IR) methods in today's information environment is pivotal to enabling effective, efficient search of distributed document collections. Recent research results in the IR literature [1, 2, 7, 8, 9] illustrate that IR researchers are recognizing and attacking this problem on a number of fronts.

It seems clear that the dynamic, distributed environment of the present and future Internet requires some changes in the way we look at searching document collections. As identified in [2], several interesting research questions arise:

- Collection Selection / Resource Discovery;
- Efficient Search; and
- Combining Search Results / Collection Fusion.

At the University of Virginia, we are particularly interested in the maintenance of *collection wide information (CWI)* (e.g. the idf and dictionary) when the collection itself is composed of separate, distributed sub-collections and where each sub-collection changes over time. Our essential thesis is that strict maintenance of CWI is costly in the loosely coupled Internet environment and is often unnecessary from an effectiveness standpoint. Our experiments to date have been centered around verifying this thesis as well as identifying how "out-of-date" CWI can be while still maintaining effectiveness commensurate with a central, static archive composed of the current corpus.

As is true for many researchers, our impetus in getting involved with TREC was to gain access to very large document collections. Our early results with small collections [7] were promising but exposed some questions that we felt could only be answered by repeating our experiments on much larger document collections.

In this paper we proceed as follows. In Section 2 we describe our general experimental framework and parameters and their embodiment in a set of software tools we call DRIFT. We also describe our TREC runs in this section. In Section 3 we present preliminary results from the Category B runs. We continue with a discussion of our results and their relation to our previous work. We conclude with lessons learned and a short description of possible future work.

2 Algorithms and Methodology

Before explaining the specifics of our experiments, we first explain the technique we use to model "out-of-date" views of collection statistics. We then outline the specific work we did on the Category B subset of the TREC corpus.

2.1 Background

2.1.1 Notation

The distributed archive is composed of s sites. The j-th document at site i is denoted by D_{ij}. At any site i, there are two collections represented. C_i^l represents the ordered collection of documents physically stored at site i - the "local" collection. The order corresponds to the insertion order of documents at that site. C_i^g represents the collection of documents that has been used to generate site i's version of CWI. We call this version G_i, so $G_i = f(C_i^g)$.

2.1.2 Dissemination of Collection Statistics

Let $prefix(d, C_i^l)$ be the first d-th fraction of C_i^l. The parameter d defines the degree of dissemination of CWI in the archive. At any point in time, site i knows about all of its own documents plus $prefix(d, C_j^l)\ \forall j \neq i$. That is

$$C_i^g = C_i^l \cup \left(\bigcup_{j \neq i} prefix(d, C_j^l) \right) \tag{1}$$

Note the following about the dissemination parameter

- d varies continuously between 0 and 1.
- When $d = 0$, no dissemination occurs and G_i is derived solely from local holdings.
- When $0 < d < 1$, G_i is derived partly from local holdings and partly from documents held elsewhere.
- When $d = 1$, complete dissemination occurs. Every site has "perfect" knowledge of every other site. So, $G_i = G_j \forall i, j$.

2.1.3 Allocation Model

One of our early findings was that the content-based allocation of documents to sites had an influence on effectiveness [7]. In particular, the more "skew" there was between the content of the collections at each site, the more dissemination was needed to achieve a specified effectiveness level. We found analogous behavior attributable to dynamism [8].

To gain parametric control over this notion of content-skew, we define an affinity probability which attempts to group content-similar documents at the same site. Our approach assumes that documents that are relevant to the same query are similar in content. We assign each query Q, a random home site, $QHome(Q)$. Documents are then assigned to sites based on three pieces of information:

- relevance information,
- $QHome()$,
- the affinity probability a.

If document D is relevant to query Q, then D is assigned to $QHome(Q)$ with probability a, and is assigned at random across all sites with probability $1 - a$. When $a = 0$, documents are assigned at random to sites in the distributed system without regard to their content. As a approaches 1, documents relevant to the same query are increasingly likely to be colocated. When $a = 1$, content-skew is maximal. Of course this methodology is limited to testing environments in which relevance judgements are available.

2.2 Experimental Framework

2.2.1 DRIFT

The software we use to run our dissemination and allocation experiments is called DRIFT. DRIFT is an object-oriented implementation of the Vector Space Model [5] written in C++ and designed specifically to perform experiments in distributed IR.

The DRIFT architecture is shown in Figure 1A, and several functions are illustrated in Figure 1B-D. In keeping with the object oriented paradigm, the DRIFT architecture can be viewed as a series of communicating objects. The fundamental objects in DRIFT are *Documents*, *Collections*, *Searchers*, *Sites*, *DocStreams*, and *CWIs*. There is also a distinguished object called the *Master* that controls the retrieval experiment.

A *Site* maintains a *Collection* (a group of *Document* objects), a *CWI*, and a *Searcher*. The *CWI* maintains the local view of the collection wide information while the *Searcher* maintains a list of term weighting strategies and the machinery to run searches with these strategies.

The *DocStream* object serves as a source of documents to the IR system. In our architecture, documents from the *DocStream* are channeled through the *Master*. The *Master* decides where the document should reside based upon the allocation policy in place and then sends the document to that site. This is not necessarily the insertion mechanism one would expect in an operational distributed IR system, but it is convenient for research purposes. The source for a *DocStream* can be any collection of documents, though in order to perform evaluation experiments, the collection must have an accompanying set of queries and relevance judgements. Document insertion is depicted in Figure 1B.

When searches need to be conducted (Figure 1C), the *Master* broadcasts the query to each *Site* (Step 1). The *Searcher* at the site then runs the search using the local *CWI* (Step 2) and returns the results to the *Master* for collation with results from other sites (Step 3).

For maximal control over the system, dissemination activities (Figure 1D), are initiated by the *Master* by sending a simple "disseminate" message to each *Site*. Each *Site* then sends the documents on its dissemination list (including documents it received from other *Sites*) to its neighbor *Sites*. The contents of this list are determined by the local dissemination policy, which in turn is configurable on a site-by-site basis. We currently fix both dissemination policy and level to the same values at all sites. Documents received from places other than the *Master* are used to update the local *CWI* object but are not actually inserted in the local collection.

Currently, DRIFT does not maintain an inverted index - all searching is done using the query and document vectors directly. In our small collection experiments [7], inverted indexes were unnecessary as we were able to achieve acceptable performance without them. However, with the large size of the TREC corpus, this initially convenient implementation decision severely constrains the scope of our work. We will have to add inverted index based searching in the future.

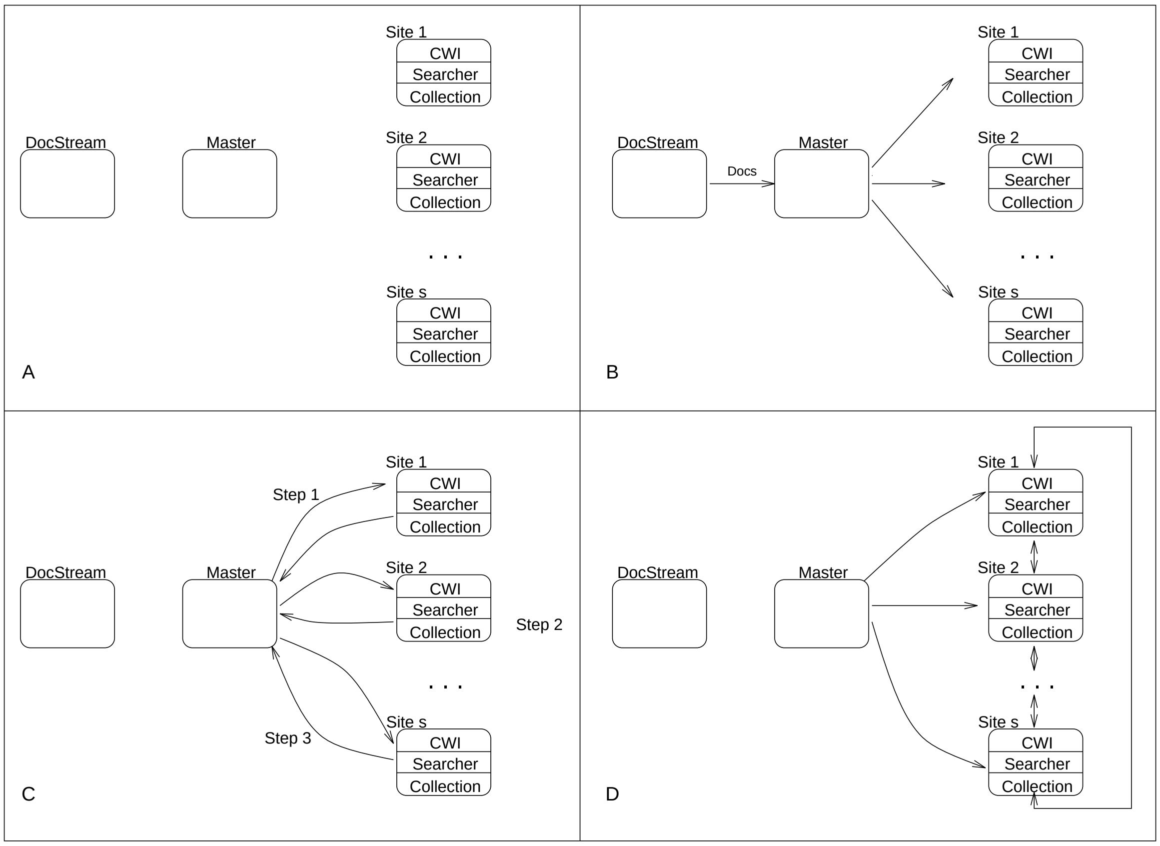

Figure 1: DRIFT architecture. The overall object structure is depicted in (A). Interactions for insertion (B), search (C), and dissemination (D) are also shown.

In Figure 2, we illustrate how document collections are constructed and processed in DRIFT. The raw text of the collection is given as input to the unmodified SMART v11.0 software. SMART removes stopwords, performs word stemming, and produces the dictionary and term frequency based document and topic vectors. We then take these vectors and convert them to a format that is suitable to the access patterns typical of our environment.

Once the collection is constructed, we use the DRIFT format and a parameter file to perform multiple runs for a particular experimental configuration. A configuration consists of values for number of sites, allocation, dissemination, document term weighting, query term weighting and other parameters. Over the life of a suite of experiments, we typically fix all parameters except the allocation and dissemination levels. Multiple runs are necessary because of the stochastic component of allocating documents to sites.

The output from the multiple runs is run through `trec_eval` and a perl program to generate mean and standard deviation estimates for several IR effectiveness measures.

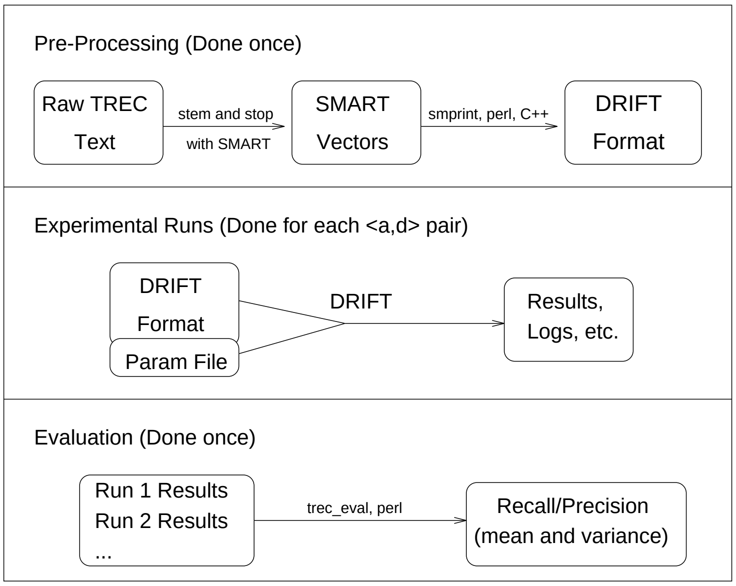

Figure 2: Collection Construction and Processing in DRIFT.

2.2.2 TREC runs

Both document and query construction were completely automatic. Using the notation from Salton and Buckley [4], we used *tfc* weighted documents (i.e. cosine normalized) and *nfx* weighted queries. We made no attempt to determine the best absolute term-weighting strategies for these collections. Collection statistics for both *nfx* and *tfc* weighting are derived from the current site and that site's generally incomplete knowledge of the holdings of other sites.

Our two official runs, *drift1* and *drift2*, represent a centralized system that uses full knowledge of the collection and a distributed system of 20 sites where each site has knowledge of only its local holdings ($d = 0$) and where documents were randomly allocated to the sites ($a = 0$). After discussions with other investigators at TREC-4, we made an additional run using length normalized document term weights rather than cosine normalized weights. This run, *drift3*, is unofficial but we include it to verify that our absolute performance could be improved. The rationale for length normalization is described in Singhal *et al.*[6].

Experiments were run on a Sun SPARCserver 20 with 128 MB memory, a 50 MHz processor, and an attached 2 GB disk. The CPU for the machine was shared with other compute intensive jobs but the disk was dedicated to the TREC work.

3 TREC-4 Results

Construction of the DRIFT format collection from the raw TREC text took about seven hours. This only needs to be done once per collection. The DRIFT format collection has about 40% of the space requirement of the raw text. Each run through the Category B data using the test topics took between four and six hours depending upon the processor load.

The results of our official and unofficial TREC-4 runs are in Table 1. Of most interest to us is not absolute performance but the relative performance between the Central (C) and Distributed (D) runs.

	TREC Category B Runs			
	Official			Unofficial
Level	Drift1(C)	Drift2(D)	Δ (%)	Drift3
0.0	0.476	0.469	-1.5	0.680
0.1	0.289	0.290	+0.3	0.449
0.2	0.235	0.213	-9.4	0.308
0.3	0.164	0.153	-6.7	0.241
0.4	0.085	0.086	+1.2	0.171
0.5	0.068	0.066	-2.9	0.140
0.6	0.046	0.046	0.0	0.096
0.7	0.028	0.028	0.0	0.059
0.8	0.005	0.005	0.0	0.036
0.9	0.002	0.002	0.0	0.003
1.0	0.001	0.001	0.0	0.002
Average (non-interp)	0.109	0.105	-3.7	0.172

Table 1: TREC results on the Category B data. Central (C) and Distributed (D) results are shown for two official runs and one unofficial run.

For comparison with TREC-4 we present our results from some small collection experiments in Table 2. The important information in this table is the consistent drop in effectiveness we observed for the random-content, no-dissemination distributed archive when compared to the central archive. This drop was not observed in the TREC runs.

4 Discussion

The choice of the parameters for our two official TREC runs was motivated by our observations on small collections. For two of the four small collections we had used (MED and CACM), we had noticed effectiveness degradation in randomly allocated ($a = 0$) collections with no dissemination. We had posited that the overall *proportion* of documents that was reflected in a site's view of CWI was directly related to effectiveness. It now appears that this hypothesis is incorrect. The TREC-4 results, as well as other experiments [3], indicate that it is not the overall proportion of documents but some minimal *unbiased* sample size that is important in achieving effectiveness commensurate with centralized archives. As shown in Table 3, in the small collections, experiments with 20 sites yielded average sub-collection sizes in the 50–160 document range. Using the Category B data, average sub-collection size was about 8200 documents.

	MED			CACM		
Level	Central	Distributed	Δ (%)	Central	Distributed	Δ (%)
0.0	0.923	0.955	+3.5	0.718	0.670	-6.7
0.1	0.837	0.837	+0.0	0.662	0.586	-11.4
0.2	0.763	0.768	+0.7	0.538	0.484	-10.0
0.3	0.708	0.697	-1.3	0.472	0.409	-13.3
0.4	0.659	0.639	-3.0	0.390	0.338	-13.3
0.5	0.566	0.548	-3.2	0.325	0.276	-17.8
0.6	0.487	0.442	-9.2	0.251	0.217	-13.5
0.7	0.425	0.386	-9.2	0.175	0.142	-18.9
0.8	0.348	0.312	-10.3	0.148	0.115	-22.3
0.9	0.242	0.210	-13.2	0.093	0.070	-24.7
1.0	0.100	0.089	-11.0	0.075	0.053	-29.3
Average (non-interp)	0.537	0.523	- 2.6	0.332	0.294	-11.4

Table 2: Previous results using small collections.

Collection	Size	Sub-collection Size	Avg Precision Δ (%)
MED	1033	52	-2.6
CACM	3204	160	-11.4
TREC Category B	164777	8239	-3.7
WSJ91	42702	2135	-0.4
AP89	84728	4236	+0.7

Table 3: Summary of average subcollection sizes and the difference in non-interpolated average precision between the central site and a 20 site system using no dissemination and random document allocation.

The nature of the TREC tests precluded us from using relevance judgements to skew the allocation of documents to sites as we have done previously [7]. With relevance judgements now available, we can proceed with this work.

In absolute terms, DRIFT was the poorest performer among the Category B participants. Our post-conference failure analysis attributes this to two factors. First, we used sub-optimal term weighting functions for the TREC data. Singhal *et al.*[6] clearly shows that using cosine normalized term weights in the TREC collections yields drastically poorer effectiveness compared to term weights based on length normalization. The second factor was a bug in our query processing that effectively added a random term from another query to every query in the test set. Given the short nature of the queries, the extra term is likely to have degraded effectiveness. The unofficial run (*drift3*) that we ran after fixing these problems yielded effectiveness on par with the other Category B systems.

As is true for many TREC participants, we experienced non-trivial growing pains when modifying our software to handle large collections. The pre-TREC version of DRIFT made massive use of in-core memory and our first revision of the software attempted to address this problem by turning to disk-based structures for the document collection itself. Memory use is still high, as we continue to maintain memory resident versions of collection statistics at each site in the simulated distributed system.

Currently the documents are stored sequentially in vector format on disk. Each query for each run requires a sequential scan through the vectors. To do further experiments on the entire TREC corpus, we really need to implement inverted index structures for the documents. This is especially true since we must do multiple runs through the corpus to identify the variation in effectiveness due to the random nature of the allocation of documents to sites.

5 Summary and Future Plans

The main contribution of the TREC corpus to our work has been to clarify some ambiguous results we had observed using small test collections and to highlight bottlenecks in our retrieval system. It has also given us several directions for future study.

Our short term plans include software modifications to DRIFT in order to improve our throughput on large collections. Once this is done, we can conduct full scale experiments to see how varying content-skew (through the allocation parameter) influences effectiveness. These modifications should also enable us to improve the absolute effectiveness of the system.

In the longer term, we would like to look at the temporally ordered collections (e.g. AP, WSJ, SJMN) to characterize the drift in collection content over time, perhaps deriving CWI from subcollections defined by temporal windows rather than the extant corpus. The applicability of TREC topics for examining this issue is undetermined.

6 Acknowledgements

We thank Donna Harman for encouraging us to re-examine our system with a fresh sense of skepticism. This work was supported in part by NASA GSRP Fellowship NGT-51018 and by NASA CESDIS Grant 5555-25.

References

[1] Nicholas J. Belkin, Paul Kantor, Edward A. Fox, and J. A. Shaw. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Information Processing and Management*, 31(4):431–448, 1995.

[2] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval (SIGIR95)*, pages 21–29, Seattle, WA, 1995.

[3] James C. French and Charles L. Viles. Ensuring Retrieval Effectiveness in Distributed, Digital Libraries. *To Appear in Journal of Visual Communication and Image Representation*, 1995.

[4] Gerard Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[5] Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[6] Amit Singhal, G. Salton, Mandar Mitra, and Chris Buckley. Document Length Normalization. Technical Report CS-95-1529, Department of Computer Science, Cornell University, 1995.

[7] Charles L. Viles and James C. French. Dissemination of Collection Wide Information in a Distributed Information Retrieval System. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval (SIGIR95)*, pages 12–20, Seattle, WA, July 1995.

[8] Charles L. Viles and James C. French. On the Update of Term Weights in Dynamic Information Retrieval Systems. In *Proceedings of the 4th International Conference on Knowledge and Information Management*, pages 167–174, Baltimore, MD, November 1995.

[9] Ellen Vorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning Collection Fusion Strategies. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval (SIGIR95)*, pages 172–179, Seattle, WA, 1995.