Retrieval and Richness when Querying by Document

Eugene Yang
Georgetown University
Washington, DC, USA
eugene@ir.cs.georgetown.edu

David D. Lewis
Cyxtera Technologies
Dallas, TX, USA
desires2018paper@davelewis.com

Ophir Frieder, David Grossman
Georgetown University
Washington, DC, USA
{ophir,grossman}@ir.cs.georgetown.edu

Roman Yurchak
Symerio SAS
Paris, France
roman.yurchak@symerio.com

ABSTRACT

A single relevant document can be viewed as a long query for ad hoc retrieval, or a tiny training set for supervised learning. We tested techniques for QBD (query by document), with an eye toward their eventual use in active learning of text classifiers in a legal context. Richness (prevalence of relevant documents) varies widely in our tasks of interest. We used 658 categories from the RCV1-v2 collection to study the impact of richness on QBD variants supported by Elasticsearch. BM25 weighting on full query documents dominated other methods. However, its absolute and relative effectiveness depended strongly on richness, raising broader questions about common test collection practices. We ported Elasticsearch's version of BM25 to the machine learning package scikit-learn and we discuss some lessons learned about the replicability of retrieval results.

KEYWORDS

ranked retrieval, text classification, prototype classifiers, nearest neighbor, evaluation, TAR 2.0, e-discovery, replication in science

1 INTRODUCTION

Having a relevant item in hand, and desiring to find others, is a common information access task. A Query By Document (QBD) functionality, sometimes referred to as More Like This, Related Documents, Similar Documents, or Recommendations is common in both standalone search software and as search functionality in other applications [21].

Our interest in QBD, however, comes from another direction. In legal applications such as electronic discovery and corporate investigations, active learning [26] is used for both supervised learning of text classifiers and for machine learning-supported interactive annotation of datasets (finite population annotation or FPA) [5, 6, 9, 48]. Iterative relevance feedback (training on top-ranked documents) [40] is the most widely used version of active learning. This holds particularly for FPA in the law, where iterative relevance feedback is sometimes known as Continuous Active Learning or CAL[1] [10].

When, as is common, the first training batch consists only of a single positive example, many supervised learning algorithms either have trivial behavior (memorizing the example) or fail completely. While the second and later rounds of iterative training may compensate for inadequacies in the first round, this will not happen if a dissatisfied user abandons the process after seeing the first round ranking. QBD provides a plausible alternative to supervised learning on the first round.

One challenge in our applications of interest, however, is richness, i.e., the proportion of responsive documents in the collection. For electronic discovery in litigation contexts, richness for responsive documents routinely ranges from 50% or more, to well below 1%. For investigatory searches (such as for insider threats, sexual harrassment, or fraud) richness can be arbitrarily low. Yet there has been no systematic research into the impact of richness on QBD, and rather little on ad retrieval. This paper addresses that gap.

2 PRIOR WORK

Query By Example (QBE) capabilities have been explored for a range of data types, including text (see below), database records [52], voice [25], music [12], images [30], and video [23]. We use the term Query By Document (QBD) in discussing QBE where the query is an entire document [49].

Early work on QBD for text treated it as relevance feedback [1, 46]. QBD is more difficult, however, since relevance feedback has available both a query *and* at least one sample document [49]. The query terms provide both additional content and a form of regularization, which is particularly critical with small training sets [33]. QBD is likewise more difficult than ad hoc retrieval. Even when consisting of user-selected terms, verbose queries typically provide poorer effectiveness than shorter ones [7, 16]. QBD involves not just verbose queries, but ones that have not benefited from user term selection.

We distinguish QBD from near-duplicate detection [8], plagiarism detection [29], and related tasks. While some of the same techniques are used, in QBD the goal is retrieval of documents with related meaning, not just documents that have an edit-based historical connection.

The largest body of QBD research is in evaluation campaigns for the patent domain. Both patent applications and patent documents are used as queries to search patents, technical articles, and portions

[1] Grossman and Cormack have filed trademark applications for these terms.

DESIRES 2018, August 2018, Bertinoro, Italy

thereof [13, 31, 34, 44]. This literature almost uniformly assumes that query reformulation (particularly query reduction by dropping most terms) is necessary [44]. The same assumption is found in most non-patent QBD work [49].

Only a few studies used full documents, or at least sections, as queries [14, 47]. Of these, only one, to our knowledge, compared basic retrieval models, finding BM25 dominated other methods [14]. Query reduction has obvious benefits for *efficiency* in querying an inverted file retrieval system. However, no such efficiency benefit exists with typical machine learning software architectures, motivating us to look at full document querying more systematically.

3 RICHNESS, RETRIEVAL, AND QBD

Past research on ad hoc retrieval, including QBD, has in two ways assumed a narrow range of richness values.

First, many ad hoc retrieval algorithms make modeling assumptions that imply low richness. Probabilistic retrieval methods, including BM25, derive inverse document frequency (IDF) weights from the assumption that the collection contains no relevant documents [11, 35, 36]. Many language modeling retrieval approaches treat a query as generated by single draw from a mixture distribution over documents [51]. This is equivalent to assuming that there is a single relevant document in the collection.

Second, a narrow range of richness values is usually imposed in test collection construction. Queries with too low or too high richness are typically discarded, and no more than a few thousand documents are assessed for queries [18]. Richness of documents coded relevant is thus forced to fall in a narrow range, while the actual richness with respect to the simulated user need typically remains unknown. The patent collections used in QBD studies have a similar problem, given their use of patent citations (which are deliberatedly bounded and incomplete) as ground truth.

The one exception is test collections produced for research on electronic discovery in the law. Some of these collections have purportedly complete relevance judgments [15] or stratified samples that allow rough estimates of richness [22]. Further some topics have relatively high richness. The number of such topics in these collections is quite small, however, and no studies of the impact of richness in these collections has been made.

The situation is very different in supervised learning. Commonly available test data sets for machine learning vary widely in class imbalance (richness in the binary case), and the impact of class imbalance on classification has been the subject of much research [20, 24]. The impact of class imbalance on common supervised learning algorithms is despite the fact that most of these methods treat the two classes (in a binary problem) symmetrically. One might expect the degree of class imbalance, i.e. richness, to have an even stronger effect on ad hoc retrieval methods since these methods do make assumptions about richness and asymmetry.

The possible importance of richness *was* anticipated in some very early work on ad hoc retrieval. Salton studied the impact of generality (what we call richness here) on precision, finding that precision decreased with generality [41, 42]. These results, however, were based on either (a) comparing different collections of documents, or (b) altering the definition of relevance (e.g. number of agreeing assesors) on a single collection. Both these approaches introduce conflating factors that make interpreting Salton's results unclear. Further, the collections used were tiny by modern standards (at most 1400 documents), and richness variations were not large (at most a factor of 7). Lewis and Tong drew on Salton's results in studying the impact of text classification components on information extraction systems, but did not examine ad hoc retrieval [27].

As a terminological matter, Robertson defined "generality" to have the same meaning we give "richness" here [37]. The term generality, however, has been used ambiguously in the information retrieval literature, however, and is more commonly used to refer to breadth of meaning. We therefore use the term richness, which has emerged in e-discovery.

4 METHODS

Our interests in QBD, the impact of richness, and in adapting ad hoc retrieval methods for supervised learning were reflected in our methodological choices.

4.1 Dataset

Our experiments used the RCV1-v2 text categorization test collection [28]. The collection contains 804,414 documents that have been completely assessed for 823 categories from three groups (Topics, Industries, and Regions). We used the 658 categories with 25 or more relevant documents.

As QBD queries for each category, we selected 25 documents by simple random sampling from that category's positive examples. The definition of relevance for each query was membership in the category, and thus richness was simply the proportion of the collection labeled with that category. Document vectors were prepared using the original XML version of each document[2]. We extracted text from the title, headline, dateline, and text subelements, concatenating them (separated by whitespace) for input to tokenization (discussed below).

No stop words, stemming, phrase formation, or other linguistic preprocessing was used. This reflected our interest in applying QBD techniques in machine learning systems that may not include a broad range of text analysis options.

4.2 Software

We present ad hoc retrieval results produced using two open source software packages. One was Elasticsearch, a distributed search engine built on top of Lucene. We used version 6.2.2 which incorporates Lucene 7.2.1. Our second set of results was produced using version 0.19.1 of scikit-learn, an open source package for machine learning, along with our modifications to that code.

Some care was required to compare results from these two systems. Supervised learning software is designed to apply a model to every object of interest (documents for us). Every documents gets a score, but the system is silent on ranking and tiebreaking. Search software, on the other hand, guarantees a ranking (total order) for retrieved documents, using implicit tiebreaking when scores are tied. However, only a subset of documents may get scores, and even fewer may be retrieved and ranked.

[2] http://trec.nist.gov/data/reuters/reuters.html

To allow comparison, we forced Elasticsearch to retrieve all documents that had any term in common with a query, by setting the `index.max_result_window` to a number that is larger than the size of collection. We output the resulting scores and document IDs, then assigned a score of 0 to all unretrieved documents. For scikit-learn, we took the dot product of each document vector with the query vector, thus producing a score (possibly 0) for each document. Then for both Elasticsearch and scikit-learn runs, all documents were sorted on score, with ties broken using the MD5 hash of the document ID. Elasticsearch's implicit tiebreaking was not used. The total orderings were input to our evaluation routines.

Experiment framework and scripts are published on Github[3] for replicability.

4.3 Evaluation

A single run applied an ad hoc retrieval algorithm on a QBD query to produce a total ordering of the collection. Since the query itself is a document in the collection, we used residual collection evaluation [17], omitting the query document from the ranking before evaluation. Thus each query is evaluated on a collection of 804,413 documents from which only itself is omitted.

We chose residual precision @ rank k ($P@k$), for values of k from 1 to 20, as the primary effectiveness measure. This reflects the use of QBD in interactive systems, including iterative relevance feeedback approaches to active learning, where the top of the ranking is of primary concern. We also computed residual R-precision, i.e., $P@k$ where k is one less than the number of relevant documents for that category. The latter measures the ability of a method to achieve high recall with QBD.

Our interest was less in any particular query document, than in the overall difficulty of QBD on that category. Therefore, for each value of k, we averaged the $P@k$ values across the 25 query documents for each category to get a category-level average.

Then to summarize the impact of richness on effectiveness, we further took the mean of category-level average effectiveness across groups of categories with similar richness. These *richness bins* were formed by rounding the logarithm (base 2) of richness to the nearest integer, and grouping together all categories that rounded to the same integer. The frequencies of our categories ranged from 0.465 (bin number -1) to our enforced lower cutoff of 25/804414 = 0.00003 (bin number -15). To ensure a minimum of 50 categories per bin, bins -1 to -6 were combined into a single bin, as were bins -14 and -15.

5 ELASTICSEARCH EXPERIMENTS

Elasticsearch is widely used, open source, and provides explicit support for QBD. It was therefore a natural choice for our first experiments.

5.1 Retrieval Methods

Elasticsearch supports QBD through its More Like This (MLT) option. MLT converts a query document to a disjunctive (OR) query using (by default) up to 25 terms from the query document. Also by default, a term must occur at least twice in the query document to be selected, and must occur in at least five documents in the collection (including the query document). Terms that satisfy these criteria are ranked by TFxIDF value (see description of vector space retrieval below) and by default the top 25 are selected. We retained all default settings in our experiments.

[3] https://github.com/eugene-yang/DESIRES18-QBD-Experiments

> USA: U.S. weekly cash lumber review - April 4.
> U.S. weekly cash lumber review - April 4.
> Random Lengths Gross List Cash Lumber Prices Quotes: (2x4 Std&Btr) WEEKLY MIDWEEK PREV WK YR AGO Inland Hem-Fir 450 440 435 - Southern Pine Westside 470 470 470 - Western Spruce-Pine-Fir 390 383 372 - Framing Lumber Composite 441 - 432 354 COMMENT - Trading started the week on a fairly active note and then picked up steam when it became clear there was no Canadian "wall of wood" waiting to cross the border at the start of a new quota shipments year on April 1, Random Lengths said. Prices slanted upward, with momentum gaining toward week's end. Apart from a nasty snowstorm in the Northeast, improving weather across the country boosted outbound shipments from dealer yards. Many buyers expressed surprise, and a little frustration, with how quickly mills cleaned up floor stock and extended order files to mid-April or beyond. While dealers stuck mostly to highly specified truckload purchases, wholesalers and distributors showed more willingness to own wood. Secondaries who took small long positions turned them fairly quickly as some dealers scrambled to cover needs they had delayed purchasing earlier. The need for stock put a premium on prompt shipments, favoring distribution yards and reloads. ...

Figure 1: A Portion of a Sample Query Document

For comparison, we also formed disjunctive queries from query documents by taking the OR of all terms in the query, and executed those queries using the Simple Query String (SQS) option. The only difference between the MLT and SQS runs were that the SQS runs used all terms in the query document, while MLT runs uses a subset of those terms. Figure 1 shows of a portion of one RCV1-v2 document. The corresponding full document retrieves 803,940 documents when a disjunctive query is formed from all terms, but only 204 documents when MLT is used to produce a reduced query.

MLT and SQS queries both support ranking the retrieved documents using any of several ad hoc retrieval methods. We tried one version each of Okapi BM25 probabilistic retrieval [38, 39], vector space retrieval [43], and language model based retrieval [45, 50, 51].

For BM25 a document is a vector of saturated TF weights produced by a function that incorporates document length normalization [36]. We used the default Elasticsearch values of b=0.75 and $k1$=1.2 for the BM25 parameters. A BM25 query is a vector of probabilistic-style IDF weights, optionally multiplied by within query weights [36]. Elasticsearch uses raw query term frequency weighting by default and we retained that.

For VSM retrieval, document vectors are produced by multiplying a TF (within document term frequency) component and an IDF component for each term, and then normalizing for document length. We used raw term frequency, the smoothed IDF version

provided by Elasticsearch (Section 6.2), and L2 document length normalization. L2 normalization (also known as cosine normalization in information retrieval) divides the TFxIDF weights by the square root of the sum of their squares, giving a Euclidean (L2) norm of 1.0 for all document vectors. This is the `classic` retrieval model in Elasticsearch.

For LM based retrieval we used Dirichlet smoothing with μ = 2000, the default value from Elasticsearch.[4]

5.2 Results and Discussion

Table 1 shows the macroaveraged P@5 and R-Precision values for the retrieval models and query variants in Elasticsearch.

For every richness bin and retrieval model, queries based on a subset of terms (Elasticsearch MLT) were no better (and usually worse) than using all terms from the QBD document in the query. This contradicts assumptions in the QBD literature that query reduction is critical for good effectiveness. Admittedly, the query reduction method implemented in Elasticsearch is less sophisticated than some proposed in the research literature, and may be motivated more by its impact on efficiency than on effectiveness. That said, Elasticsearch is widely used, often with defaults unexamined and unchanged.

Both VSM and LM retrieval are based on a notions of query/document similarity, with the latter treating documents as probability distributions from which the query might be generated. A view of retrieval as similarity might seem natural for QBD, and even more so in our simulated setting where query documents are selected at random rather than by a user with intention.

It is notable, therefore, that Elasticsearch's BM25 default model dominates its VSM and LM default models for almost all richness conditions and both query forms. There have been numerous published language model variants, and it is plausible that one would do better than BM25 on our dataset, and on QBD in general. But our results, at least, cut against simplistic notions that QBD is simply similarity matching.

Our most striking result was the sharp, nearly monotonic decline in absolute effectiveness with declining richness for all retrieval models and both query types. The monotonic decrease in effectiveness appears not only for the recall-oriented R-precision metric, but even for $P@5$ (and all choices of $P@k$ for k = 1 to 20).

Figure 2 shows a scatter plot of richness vs $P@20$ for BM25 similarity using the full disjunctive (SQS) query on all 658 categories. The plot shows the strong correlation between richness and effectiveness, though also a substantial category-to-category variation.

The Pearson (linear) correlation between the base 2 logarithm of richness and residual P@20 is 0.58.[5] For comparison, the maximum Pearson correlation of 22 query effectiveness prediction methods studied by Hauff, Hiemstra, and de Jong was 0.52 across a set of datasets [19]. Richness with respect to a query is of course not known in operational settings, so richness is not a practical pre-retrieval effectiveness prediction cue. What this comparison does

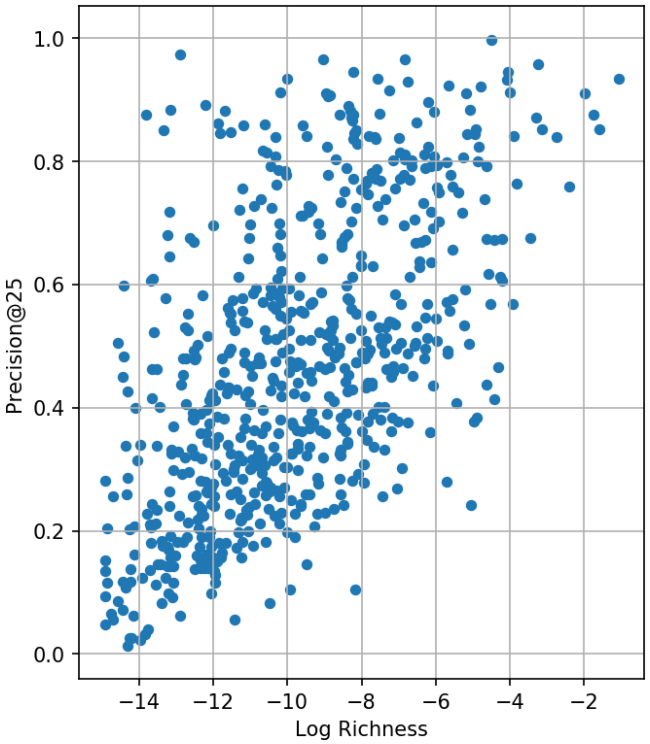

Figure 2: Relationship between category richness and mean (over 25 QBD queries) of Precision@20 for 658 RCV1 categories. Retrieval uses a disjunction over all terms in a query document with ranking by Elasticsearch's implementation of the BM25 retrieval model. Pearson correlation is r = 0.58.

show is the power of richness as a confounding factor in studying effectiveness, compared to other query characteristics that are commonly viewed as important.

Most IR researchers and practitioners would agree that low richness makes retrieval more difficult. Yet, the routine success of web search engines at very low richness tasks has perhaps led to a certain complacency. It is notable that overviews of the TREC HARD track (which focused on retrieval for difficult ad hoc queries) do not even mention richness as a contributor to low effectiveness[2–4].

Richness also affects relative effectiveness, though less strongly. The rank ordering of the six methods is largely the same across richness bins. However, the relative difference between the best and worst P@5 scores for the six methods increases from 6% for the highest richness bin to 15% for the lowest richness bin. Thus, as richness decreases the choice of retrieval method becomes more consequential. This has obvious implications for trading off cost versus complexity in operational systems.

We focused on $P@5$ in our analysis to reflect our interest in QBD for interactive interfaces. Five documents is close to the maximum within which a user can immediately perceive the presence of relevant documents. The $P@k$ results for all depths from 1 to 20 follow a similar pattern, however, with relative differences being more stable across richness bins as k increases. The R-precision results, which correspond to $P@k$ for k equal to the number of documents, are a limiting case.

[4] https://www.elastic.co/blog/language-models-in-elasticsearch
[5] We used $P@20$ instead of $P@5$ for increased granularity in the figure, but correlations are similar at all depths.

Table 1: QBD effectiveness for 6 adhoc retrieval variants. Effectiveness was averaged first across 25 random QBD queries per category, and then across categories falling in the same richness bin. Binning was on a logarithmic (base 2) scale, from 2^{-6} (and above) to 2^{-14} (and below).

			Richness Bin and Number of Categories in Bin								
			≥ -6	-7	-8	-9	-10	-11	-12	-13	≤ -14
	Query	Model	86	54	78	70	90	81	85	59	55
Precision @ 5	MLT	BM25	0.81	0.67	0.66	0.60	0.58	0.52	0.47	0.47	0.35
	MLT	LM	0.77	0.63	0.62	0.56	0.54	0.49	0.44	0.45	0.33
	MLT	VSM	0.80	0.63	0.63	0.58	0.55	0.50	0.43	0.45	0.31
	SQS	BM25	0.81	0.67	0.67	0.61	0.61	0.54	0.50	0.50	0.37
	SQS	LM	0.78	0.64	0.64	0.58	0.58	0.51	0.47	0.47	0.35
	SQS	VSM	0.80	0.65	0.66	0.60	0.58	0.53	0.48	0.48	0.35
R-Precision	MLT	BM25	0.28	0.18	0.19	0.17	0.14	0.14	0.12	0.16	0.15
	MLT	LM	0.26	0.16	0.18	0.16	0.14	0.14	0.12	0.16	0.14
	MLT	VSM	0.26	0.15	0.17	0.16	0.13	0.13	0.11	0.15	0.13
	SQS	BM25	0.31	0.18	0.20	0.18	0.14	0.14	0.13	0.17	0.16
	SQS	LM	0.29	0.17	0.18	0.17	0.15	0.14	0.13	0.17	0.14
	SQS	VSM	0.31	0.16	0.18	0.18	0.15	0.14	0.13	0.16	0.15

Are the differences discussed here statistically significant? Claims of statistical significance in ad hoc retrieval experiments are typically based on the problematic assumption that the queries in a test collection are a random sample from some population. Outside of query log studies, this is always false. It is particularly false for the RCV1-v2 categories, which are part of a single indexing system. Thus, despite the fact that each value in Table 1 is based on more than 1000 data points, we eschew such claims. We believe convincing evidence for analyses such as ours requires replication across different datasets and different experiment designs.

5.3 Comparison with LYRL2004 Results

The 2004 paper by Lewis, Yang, Rose, and Li introducing the RCV1-v2 collection includes graphs showing that text classification effectiveness generally increases with increasing category richness [28]. That analysis, however, was based on classifiers produced by applying supervised learning to a training set of 23,149 documents. Category richness strongly affected the number of positive examples present for a given category within that fixed training set.

Thus the RCV1-v2 results conflated the quality of the training data available for a category with the difficulty of the classification task for that category. In contrast, our results are based on making exactly the same amount of data (one positive document) available for each run on each category.

The impact of richness was also obscured in the RCV1-v2 paper by its focus on binary text classifiers evaluated by F1 (harmonic mean of recall and precision). For categories with high richness, large values of F1 can be achieved simply by classifying all test examples as positive. For RCV1-v2, the highest richness category would get an F1 score of 0.635 under this strategy [28]. Even random classification of documents will produce a nontrivial value of F1 for high richness categories. This lower bounding of effectiveness for trivial approaches reflects the fact that richness is, for practical purposes, a floor on precision [37, 42].

Our experimental design was not immune to this generality effect [42], but minimized it to the extent possible. We ensured that effectiveness values in the full range from 0.0 to 1.0 were logically possible for all queries, categories, and measures. Thus neither absolute nor relative effects of richness result from ceiling or floor effects. Our use of ranking-based effectiveness measure means there is no analogue to a trivial system that treats all examples as relevant. For the highest frequency category, a trivial system that randomly ordered documents would have an expected precision of 0.465 for the most frequent category. However, for most categories the expected precision of such a system would be well below 0.01.

6 REIMPLEMENTATION IN SCIKIT-LEARN

A major impetus for our work is the hope of using QBD methods to improve first round effectiveness for active learning of text classifiers. As a first step, we reimplemented Elasticsearch's BM25 variant in scikit-learn, a machine learning toolkit.

We first created a matrix of raw term frequency values using CountVectorizer from scikit-learn[6]. We then extended an existing BM25 implementation[7] and created BM25 query and document vectors intended to be identical to those from Elasticsearch.

The first two lines of Table 2 compare the Elasticsearch BM25 results using all query document terms to those from our first scikit-learn implementation. To our surprse, the Elasticsearch results were notably different, and often better.

[6] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[7] https://github.com/scikit-learn/scikit-learn/pull/6973

Table 2: Compring effectiveness of Elasticsearch's BM25 retrieval on full document queries versus a converging set of implementations in scikit-learn. Effectiveness computed as in Table 1.

		Richness Bin and Number of Categories in Bin								
		≥ -6	-7	-8	-9	-10	-11	-12	-13	≤ -14
		86	54	78	70	90	81	85	59	55
Precision @ 5	Elasticsearch: Simple Query String	0.81	0.67	0.67	0.61	0.61	0.54	0.50	0.50	0.37
	Original Result: With IDF and Raw QTF	0.81	0.67	0.69	0.63	0.58	0.53	0.48	0.46	0.36
	Same Param.	0.81	0.67	0.67	0.60	0.59	0.53	0.48	0.48	0.35
	Same Param. + Sklearn IDF Smoothing	0.81	0.67	0.67	0.60	0.59	0.53	0.48	0.48	0.35
	Same Param. + ES IDF Smoothing	0.81	0.68	0.68	0.61	0.61	0.54	0.50	0.50	0.37
	Same Param. + ES IDF Smoothing + Token	0.81	0.68	0.68	0.61	0.61	0.55	0.50	0.50	0.37
R-Precision	Elasticsearch: Simple Query String	0.31	0.18	0.20	0.18	0.14	0.14	0.13	0.17	0.16
	Original Result: With IDF and Raw QTF	0.30	0.17	0.18	0.18	0.14	0.14	0.12	0.15	0.15
	Same Param.	0.30	0.16	0.17	0.17	0.14	0.13	0.12	0.16	0.14
	Same Param. + Sklearn IDF Smoothing	0.30	0.16	0.17	0.17	0.14	0.13	0.12	0.16	0.14
	Same Param. + ES IDF Smoothing	0.32	0.18	0.19	0.18	0.15	0.15	0.14	0.17	0.16
	Same Param. + ES IDF Smoothing + Token	0.32	0.18	0.19	0.18	0.15	0.15	0.13	0.17	0.16

6.1 Parameters

We noticed that Elasticsearch used b=0.75 and k1=1.2 for as their defaults, but b=0.75 and k1=2.0 were defaults in the BM25 implementation we built on. We changed our values in scikit-learn to the the Elasticsearch ones, with results shown in the row marked with "Same Param.". Our results were closer to those of Elasticsearch in most richness bins, but diverged slightly more in the lowest richness bin.

6.2 IDF

The standard definition of IDF weighting [36] is:

$$\log \frac{N}{n_j}$$

where N is the number of documents in the collection, and n_j is the number of documents that term j occurs in. Logarithms base 2, e, and 10 are all commonly used with IDF, with the choice having no effect in typical TFxIDF term weighting applications.

The scikit-learn BM25 implementation that served as the starting point for our scikit-learn work used scikit-learn's built-in IDF weighting, which is controlled by the boolean flag -smooth_idf.[8] Our original scikit-learn BM25 had IDF smoothing turned off. We tried a run (row: "Same Param. + Sklearn IDF Smoothing") with IDF smoothing turned on, and the results were closer to but not identical to those of Elasticsearch.

Disabling -smooth_idf in scikit-learn uses this version of IDF:

$$1 + \ln \frac{N}{n_j}$$

while enabling the option uses this version:

$$1 + \ln \frac{N+1}{n_j+1}$$

Neither of these is the standard definition of IDF weighting. Nor is either what the above scikit-learn documentation states is the "textbook" definition:

$$\log \frac{N}{n_j+1}$$

in what appears to be a typographical error.

Elasticsearch implements yet a fifth version, the one suggested by the probabilistic derivation of BM25 [36]:

$$\ln \frac{N - n_j + 0.5}{n_j + 0.5}$$

We added this variant to our scikit-learn implementation, and got the results shown in row Same Param. + ES IDF Smoothing.

6.3 Tokenization

At this point we had very similar effectiveness from the two implementations, but decided to shoot for an exact replication. The tokenizer in scikit-learn separates a character string into words at boundaries between Unicode word and non-word characters by applying regular expressions. Elaticsearch tokenizes text by applying the Unicode Text Segmentation algorithm specified in Unicode Standard Annex #29[9].

We therefore extracted the tokens and raw TF values for each document from Elasticsearch via the Term Vector API. We used this data to create a raw TF matrix in scikit-learn, and created BM25 query and document vectors as above. Surprisingly, the results, marked "Same Param. + ES IDF Smoothing + Token", still differed slightly from the Elasticsearch results.

6.4 Document Length

At this point we had identical raw TF vectors and identical weighting formulas. Yet when we examined individual document scores

[8] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[9] http://unicode.org/reports/tr29/

from corresponding runs, many were slightly different. Using Elasticsearch's explain API[10] uncovered the difference.

Elasticsearch inherits from Lucene a lossy compression of document lengths[11]. All document lengths from 0 to 2,013,265,944 are compressed into 7 bits using a coding table[12]. Elasticsearch's BM25 implementation uses these approximate lengths. We verified for individual documents that correcting for this removes the last difference between the Elasticsearch and scikit-learn scores. (We chose not to implement this quirky scheme in our scikit-learn code.)

7 FUTURE WORK

Querying with documents lives intriguingly at the intersection of ad hoc retrieval and supervised learning. Our results suggest that QBD using ad hoc retrieval workhorse BM25 is a solid approach. BM25 can be viewed as Naive Bayes combined with negative blind feedback [36]; so our results reinforce the usefulness of generative models with tiny training sets [32]. We plan next to compare the downstream effects on active learning of using BM25 on the first round and to test schemes for transitioning from BM25 to discriminative supervised learning when enough training data has been accumulated. The fact that Naive Bayes and logistic regression can be viewed as, respectively, generative and discriminative variants of the same model [32] may provide useful for this purpose.

The strong impact of richness on absolute and relative effectiveness is intriguing and cries out for study on multiple datasets. Immediate questions are 1) whether the effect is an artifact of using random examples as simulated QBD queries or of using text categorization topics as simulated user interests; 2) whether it carries beyond QBD to ad hoc retrieval with short queries; and 3) whether it is a function of simple frequency, topical generality, or both. Some of these factors can be explored with existing datasets, but others may require new test collection work.

8 CONCLUSION

Our interest in QBD was sparked by the problem of supervised learning on tiny training sets. However, using a document as a query is also a widely supported information access tool and, to our mind, an understudied one. BM25 weighting turns out to be an effective approach, and its origins in supervised learning (naive Bayes) suggest interesting approaches for kicking off active learning.

The assumption in the QBD literature that query pruning is crucial was not borne out by our work. Indeed, taken literally, pruning cannot actually be necessary. One can always achieve the same effect by sufficiently small weights, and this perhaps is a more natural perspective in supervised learning contexts. To the extent that pruning is desirable for efficiency reasons in inverted file retrieval, this perhaps should be treated as a query optimization issue, not a modeling one.

The efforts required to replicate the results of an open source search engine using an open source machine learning toolkit was a reminder of the range of factors that impact the effectiveness of text processing systems. Our experience provides yet more evidence that care is needed in interpreting small differences in published retrieval results, and that ongoing attention is needed to replicability in IR research.

Finally, the overwhelming impact of richness on effectiveness in our experiments was both intriguing and unsettling. Suppose such an effect were to hold not just for our simulation using a text categorization data set, but also for widely used ad hoc retrieval test collections. This would raise the possibility that the outcomes of many past experiments on ad hoc retrieval were predestined by test collection design choices.

Further, we would have no way to tell if this were true. With the exception of some work in e-discovery, all major public test collections have at least partially decoupled true richness for actual topical richness, and left topical richness unknown and not measurable after the fact. We suggest that organizers of future IR evaluations consider explicit control and measurement of richness in test collection creation.

[10] https://www.elastic.co/guide/en/elasticsearch/reference/current/search-explain.html
[11] https://github.com/elastic/elasticsearch/issues/24620
[12] https://issues.apache.org/jira/browse/LUCENE-7730

REFERENCES

[1] IJsbrand Jan Aalbersberg. 1992. Incremental relevance feedback. In *SIGIR 1992*. ACM, 11–22.
[2] James Allan. 2003. HARD track overview in TREC 2003 high accuracy retrieval from documents. In *TREC 2003*.
[3] James Allan. 2004. HARD track overview in TREC 2004 high accuracy retrieval from documents. In *TREC 2004*.
[4] James Allan. 2005. HARD track overview in TREC 2005 high accuracy retrieval from documents. In *TREC 2005*.
[5] Mossaab Bagdouri, William Webber, David D Lewis, and Douglas W Oard. 2013. Towards minimizing the annotation cost of certified text classification. In *CIKM 2013*. ACM, 989–998.
[6] Jason R Baron, Michael D Berman, and Ralph C Losey. 2016. *Perspectives on Predictive Coding and Other Advanced Search Methods for the Legal Practitioner*. ABA Book Publishing.
[7] Michael S Bernstein, Bongwon Suh, Lichan Hong, Jilin Chen, Sanjay Kairam, and Ed H Chi. 2010. Eddi: interactive topic-based browsing of social status streams. In *UIST 2010*. ACM, 303–312.
[8] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. 2002. Collection statistics for fast duplicate document detection. *TOIS* 20, 2 (2002), 171–191.
[9] Gordon F. Cormack and Maura F. Grossman. 2014. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. *SIGIR 2014* (2014), 153–162. https://doi.org/10.1145/2600428.2609601.
[10] Gordon F. Cormack and Maura F. Grossman. 2015. Autonomy and reliability of continuous active learning for technology-assisted review. *arXiv* (2015), 19.
[11] W.B. Croft and D.J. Harper. 1979. Using Probabilistic Models of Document Retrieval without Relevance Information. *JDoc* 35, 4 (1979), 282–295.
[12] Jonathan T Foote. 1997. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II*, Vol. 3229. International Society for Optics and Photonics, 138–148.
[13] Atsushi Fujii, Makoto Iwayama, and Noriko Kando. [n. d.]. Overview of the Patent Retrieval Task at the NTCIR-6 Workshop.. In *NCTIR 2007*.
[14] Julien Gobeill, Douglas Theodoro, and Patrick Ruch. [n. d.]. Exploring a Wide Range of Simple Pre and Post Processing Strategies for Patent Searching in CLEF IP 2009.. In *CLEF (Working Notes)*.
[15] Maura R Grossman, Gordon V Cormack, and Adam Roegiest. 2016. TREC 2016 Total Recall Track Overview.
[16] Manish Gupta, Michael Bendersky, et al. 2015. Information retrieval with verbose queries. *F&T in IR* 9, 3-4 (2015), 209–354.
[17] Donna Harman. 1992. Information Retrieval. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, Chapter Relevance Feedback and Other Query Modification Techniques, 241–263.
[18] Donna K. Harman. 2005. The TREC Test Collections. In *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press.
[19] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. 2008. A survey of pre-retrieval query performance predictors. In *CIKM 2008*. ACM, 1419–1420.
[20] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.
[21] Marti Hearst. 2009. *Search user interfaces*. Cambridge University Press.
[22] Bruce Hedin, Stephen Tomlinson, Jason R Baron, and Douglas W Oard. 2009. *Overview of the TREC 2009 legal track*. Technical Report.

[23] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. 2011. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41, 6 (2011), 797–819.

[24] Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent data analysis* 6, 5 (2002), 429–449.

[25] Martha Larson, Gareth JF Jones, et al. 2012. Spoken content retrieval: A survey of techniques and technologies. *F&T in IR* 5, 4–5 (2012), 235–422.

[26] David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR 1994*. Springer-Verlag New York, Inc., 3–12.

[27] David D. Lewis and Richard M. Tong. 1992. Text Filtering in MUC-3 and MUC-4. In *Proceedings of the 4th Conference on Message Understanding*. 51–66.

[28] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *JMLR* 5 (2004), 361–397. https://doi.org/10.1145/122860.122861

[29] Chao Liu, Chen Chen, Jiawei Han, and Philip S Yu. 2006. GPLAG: detection of software plagiarism by program dependence graph analysis. In *SIGKDD 2006*. ACM, 872–881.

[30] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. 2007. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition* 40, 1 (2007), 262–282.

[31] Mihai Lupu, Harsha Gurulingappa, Igor Filippov, Zhao Jiashu, Juliane Fluck, Marc Zimmermann, Jimmy Huang, and John Tait. [n. d.]. Overview of the TREC 2011 Chemical IR Track. ([n. d.]).

[32] A.Y. Ng and M. I. Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS* (2001), 841–848. https://doi.org/10.1007/s11063-008-9088-7 arXiv:http://dx.doi.org/10.1007/s11063-008-9088-7

[33] Andrew Y Ng. 2004. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *2004 ICML*. ACM, 78.

[34] Florina Piroi and Allan Hanbury. 2017. Evaluating Information Retrieval Systems on European Patent Data: The CLEF-IP Campaign. In *Current Challenges in Patent Information Retrieval*. Springer, 113–142.

[35] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *JDoc* 60, 5 (2004), 503–520.

[36] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *F&T in IR* 3, 4 (2009), 333–389.

[37] Stephen E Robertson. 1969. The Parametric Description of Retrieval Texts: Part I: The Basic Parameters. *Journal of Documentation* 25, 1 (1969), 1–27.

[38] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR 1994*. Springer-Verlag New York, Inc., 232–241.

[39] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. 109 (1995), 109.

[40] Joseph John Rocchio. 1971. Relevance feedback in information retrieval. (1971), 313–323.

[41] Gerard Salton. 1970. *The "generality" effect and the retrieval evaluation for large collections*. Technical Report 70-67. Cornell University, Ithaca, NY.

[42] Gerard Salton. 1972. The "generality" effect and the retrieval evaluation for large collections. *Journal of the Association for Information Science and Technology* 23, 1 (1972), 11–22.

[43] G. Salton, C.S. Yang, and A. Wong. 1975. A Vector-Space Model for Automatic Indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[44] Walid Shalaby and Wlodek Zadrozny. 2017. Patent Retrieval: A Literature Review. *arXiv preprint arXiv:1701.00324* (2017).

[45] Fei Song and W Bruce Croft. 1999. A general language model for information retrieval. In *CIKM 1999*. ACM, 316–321.

[46] Amanda Spink, Bernard J Jansen, and H Cenk Ozmultu. 2000. Use of query reformulation and relevance feedback by Excite users. *Internet research* 10, 4 (2000), 317–328.

[47] Suzan Verberne and Eva DâĂŹhondt. 2009. Prior art retrieval using the claims section as a bag of words. In *CLEF 2009*. Springer, 497–501.

[48] E. Yang, D. Grossman, O. Frieder, and R. Yurchak. 2017. Effectiveness Results for Popular e-Discovery Algorithms. *ICAIL 2017* (2017).

[49] Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. 2009. Query by document. In *WSDM 2009*. ACM, 34–43.

[50] ChengXiang Zhai. 2008. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies* 1, 1 (2008), 1–141.

[51] ChengXiang Zhai and John Lafferty. 2002. Two-stage language models for information retrieval. In *SIGIR 2002*. ACM, 49–56.

[52] Moshé M Zloof. 1975. Query by example. In *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*. ACM, 431–438.