WaterlooClarke: TREC 2015 Total Recall Track

Haotian Zhang, Wu Lin, Yipeng Wang, Charles L.A. Clarke and Mark D. Smucker
University of Waterloo
haotian.zhang@uwaterloo.ca, wu.lin@uwaterloo.ca, yipeng.wang@uwaterloo.ca,
claclark@uwaterloo.ca, mark.smucker@uwaterloo.ca

ABSTRACT

The total recall track in TREC 2015 seeks an enhanced model to accelerate the autonomous technology-assisted review process. This paper introduces several noval ideas such as clustering based seed selection method, extended n-grams features and continuous query expansion learned from the relevant documents derived from each iteration. These methods can retrieve more relevant documents from each iteration thereby achieving high recall while requiring less review effort.

1. INTRODUCTION

The technology-assisted review ("TAR") applies the iterative retrieval and review of documents to find a substantial majority or all of the relevant documents in a collection. Its applications include electronic discovery ("eDiscovery") in legal matters [2], systematic review in evidence-based medicine [5], and the creation of test collections for information retrieval ("IR") evaluation [10]. Since the reviewers are typically experts in the subject matter, such as lawyers and medicine specialists, assessing massive documents is extremely expensive. For this reason, it is desirable to maximize the recall and minimize the number of reviewed documemts at the same time.

According to the literature review performed by the authors, there are a number of search efforts aimed at achieving high recall, especially in the field of eDiscovery and IR evaluation. However, most of them require the help from search experts [7] or topic- or database-specific training [14, 15]. In addition, many search methods are not reliable, and require extensive efforts for some topics although the effects on average may be acceptable.

Our goal is to design a more efficient retrieval system to achieve high recall while requiring less judge effort from reviewers. Also, if possible, the system should work generally well for any topic and any collection. To the best of our knowledge, the two automatic TAR tools widely utilized by legal service providers are Simple Active Learning ("SAL") and Simple Passive Learning ("SPL") [2]. The SPL protocol constructs the training set based on the operator or random selection, while the SAL protocol uses a machine learning algorithm [11] to identify the training set and always selects documents lying closest to the decision surface, where the learning algorithm is least certain for review [12]. Cormack and Mojdeh proposed a new protocol called Continuous Active Learning ("CAL") which is similar to the tradition SAL and applies a keyword search, such as BM25, to identify an initial set of documents [4]. But it sends the top-scoring (most certain) documents identified by the learning algorithm to the reviewers. Cormack and Grossman claimed that CAL is generally more effective than the other two methods and SAL is as effective as CAL only in a best-case scenario [2].

Cormack and Grossman moved one step forward and came up with an autonomous TAR (Auto TAR) configuration that exhibits "greater autonomy, superior effectiveness, increased generalizability, and fewer, more easily detectable failures" compared to existing TAR methods [3]. This protocol is implemented as the baseline of TREC 2015 Total Recall Track and its details are discussed in the next section.

2. BASELINE MODEL

Previous researchers tried three different ways to construct the initial training set [3]. The method called "Auto-BM25" was seeded with the top-ranked documents given by BM25, while another one labeled "Auto-Syn" was seeded using a synthetic document created from the query. The last method is called "Auto-Rand" and simply selects a random relevant document at the outset. According to the test results on TREC 2009 Legal Track topics and TREC 2002 Filtering Track topics, we can conclude that the Auto-Syn generates better results than the other two, and reckon that's the reason why the Auto-syn is implemented as the baseline in this year's total recall track.

As we just discussed, the baseline constructs a synthetic document based on the topic as a first relevant document, and adds it to the training set. Then, 100 documents are randomly selected from the corpus and added to the training set as irrelevant documents. A logistic regression classifier trained by these 101 documents ranks all the documents based on how closely they are relevant to the query, and the highest-scoring documents are chosen for expert review. After that, these reviewed documents with their labels are added to the training set and another 100 random documents are brought in as irrelevant documents. The procedure repeats until all relevant documents are retrieved or some predefined criteria is satisfied. Notably, since the training set is augmented with more judged documents, the classifier becomes more accurate and it's reasonable to send more top-scoring documents for review in a later iteration. Cormack claimed that generally, all Auto TAR runs achieve moderate levels of recall with less review effort than CAL, but for very high levels of recall are indistinguishable from CAL [3]. The SAL and SPL gain curves are generally inferior.

3. IMPROVED RETRIEVAL MODEL

3.1 Clustering-based Seed Selection

We use an interactive procedure to select potential relevant documents. The first approach is inspired by the multi-armed bandit problem. This approach is called the sampling approach. The approach is listed as below:

1. Select D documents as candidate documents for seed selection;
2. Group these documents into K $(1 < K < D)$ clusters, where K can be given or learned from data;
3. Select one document from each cluster and label the document based on relevance feedback from experts;
4. Initialize cluster-specific counter t_{cluster} to 1 and cluster-specific reward r_{cluster}, based on judgments in step 3;
5. Iteration: select cluster v based on the following conditions:
   (a) at least one unlabeled document in cluster v;
   (b) the maximum value of $\frac{r_{\mathrm{v}}}{t_{\mathrm{v}}} + \sqrt{\frac{\mu \log(\sum_{c=1}^{|C|} t_c)}{t_{\mathrm{v}}}}$ among cluster set C. Note that cluster c belongs to C only when there is at least one unlabeled document in cluster c;
6. Pick a unlabeled document from cluster v and label the document based on relevance feedback from experts, which is the same as in step 3;
7. Update cluster-specific reward r_{v}, based on judgment in step 6 and increase the counter t_{v} by 1;
8. Go to step 5 until all D documents are labeled;

In the second approach, a graph is used to represent relationship among documents. This approach is called the graph approach. The main idea of the approach is:

1. Select D documents as candidate documents for seed selection;
2. Build a weighted graph for these documents based on results from clustering;
3. Iteration: select one unlabeled document d from the graph and label it based on relevance feedback from experts;
4. If document d is:
   (a) relevant, increase weights of edges connected to the document;
   (b) irrelevant, decrease weights of edges connected to the document;
5. Go to step 3 until all D documents are labeled.

In this paper, top D retrieved documents ranked by BM25 are used in step 1. Latent semantic indexing (LSI) is applied to learn the conceptual correlations of documents and is based on the entropy weighting term-document matrix [6]. Clustering is operated upon the features generated from entropy weighting LSI and is a great way to group documents based on their conceptual similarity. The graph in step 2 of the graph approach is constructed by:

1. Documents are considered as nodes in the graph;
2. We run K-means T times to cluster these documents;
3. The weight $w_{i,j}$ of a un-directed edge between node i and node j is $w_{i,j} = \sum_{t=1}^{T} \mathrm{I}_t(i,j)$,

where $\mathrm{I}_t(\cdot)$ is an indicator function and $\mathrm{I}_t(i,j) = 1$ denotes document i and document j are in a cluster based on the t-th clustering result of K-means.

The following greedy heuristic is used in step 3 of the graph approach.

1. Initialize a priority queue for documents
2. If the queue is:
   (a) empty, select a document with the highest BM25 score among all unlabeled documents;
   (b) not empty, select a document with highest weight from the queue;
3. Label document i based on relevance feedback from experts;
4. If document i is:
   (a) relevant, let $\delta(i) = 1$;
   (b) irrelevant, let $\delta(i) = -1$;
5. For each unlabeled document j which connects to document i,
   (a) if document j is not in the queue and document i is relevant, insert document j into the queue with weight $w_{i,j} + \delta(i)$;
   (b) if document j is in the queue, increase the weight of document j in the queue by $\delta(i)$.

3.2 Seed Selection Strategies Comparison

Apart form the strategy proposed in the previous section, we also tried several other strategies to select initial seed set. (1) Clustering Jumping: We select the document d_i^n (i-th document in n-th cluster) with the highest BM25 score for judging. Assuming it is coded as relevant, then a document d_j^n with second-highest score is picked up from the same cluster c_n; otherwise, we select the highest-scored document d_k^m from other cluster c_m. This procedure continues until a certain number of relevant documents are obtained. (2) Weighted Clustering: The method is similar to the Clustering Jumping and the difference between them is that an initial weight value of 1.0 is assigned to each cluster c_n. If a document d_i^n is labeled as irrelevant, the corresponding cluster c_n weight is multiplied by a heuristic factor which is less than 1. Then the document d_j^m in the cluster c_m with the highest factorized weight is chosen for next iteration.

Table 1 shows the comparison results for the 7 topics(tr0-tr7) on the oldreut and 20ng corpora provided by Total Recall organizers[1]. It should be noticed in Table 1 that the BM25 ranking only returns 7 documents for tr2 and 63 documents for tr3. So we just skip the clustering algorithm and send all the 7 documents about tr2 for review. For tr3, we only select half of the returned 63 documents for review. The Graph based method achieves the superior results compared to other methods in the most topics.

Table 1: Seed Selection Strategies Comparison Using 50 Effort

Methods	tr0	tr1	tr2	tr3	tr4	tr5	tr6
Jumping	46	1	2	10	47	49	40
Weighted	46	0	2	10	47	49	42
Sampling	45	1	2	14	**48**	49	**46**
Graph	**47**	**2**	2	**15**	45	**50**	45

3.3 Early Stop

In step 5 of the graph approach and step 8 of the sampling approach, we stop the iteration when all documents are labeled. Our experiments show that usually seed selection works well to identify relevant documents at the beginning. Since usually not all candidate documents in seed selection are relevant, our experiments also show using an early stop strategy can improve overall performance. We use the precision to measure performance in this strategy. The strategy we used is:

1. Set a width, w, of a window to watch performance of seed selection;

2. Set a lower bound of acceptable precision, r, $(0 < r < 1)$, to measure performance in a window of w iterations;

3. Set a width, u, of a window to tolerate unacceptable performance in a window of w iterations;

4. Let the seed selection program run the first w iterations, set counter c to zero, and let $t = w + 1$;

5. At t-th $(t > w)$ iteration of seed selection, let d be the number of relevant documents found at iteration interval $[t - w + 1, t]$. If $\frac{d}{w}$:

    (a) $< r$, increase c by 1;

    (b) $\geq r$, set c to zero;

6. If counter c:

    (a) $> u$, stop the seed selection program;

    (b) $\leq u$, increase t by 1 and go to step 5.

3.4 Feature Engineering

The baseline model utilizes two different features to represent each document. These features are vectorized and set as input vectors to train a logistic regression classifier. One of these features is TF-IDF word-based feature and the other one is binary byte 4-grams feature (combinations of 4 sequential characters). According to the experiment results, a method applying TF-IDF word-based feature performs better than that with binary byte 4-grams feature in most cases. However, we find that binary byte 4-grams feature achieves higher accuracy especially when the query is a complete sentence or composed of multiple words. As for keyword query, TF-IDF is effective enough to train a highly accurate linear classifier.

An intuitive idea is to combine the results derived from these two features so that the classifier can gain different information. We apply the RRF (Reciprocal Rank Fusion) fusion on the ranking lists R generated from a set D of documents [1].

$$RRF_{score}(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{1}$$

RRF fusion ensures that the highly ranked documents from both features are more important, while the lower-ranked documents does not vanish, where $k = 60$ was fixed according to previous experience. The result of this fusion is not as satisfying as we expected. It generates almost the same result as that from TF-IDF, yet consumes more computation costs.

The second idea is using the entropy g_i which describes the relative frequency of term i within the entire collection of documents. We can also use the entropy weighting LSI as features of documents. It can be defined as:

$$g_i = 1 + \sum_j \frac{p_{ij} \log p_{ij}}{\log n}, \text{where } p_{ij} = \frac{tf_{ij}}{gf_i} \tag{2}$$

where n is the number of documents in the corpus, gf_i denotes the occurrences of term i in the whole corpus and tf_{ij} indicates the term frequency of term i in the document j. LSI performs a Singular Value Decomposition (SVD) on the matrix and reduces the high dimensional sparse term-document matrix into a given size compact matrix. These two features are generated during seed selection phase and we can directly use them for classification. Although both of the features extract the most key information from each document and perform well on clustering, we find that they are not able to make the classification more precise when comparing with TF-IDF. We assume that the dimension reduction would lead to a information loss to a certain extent. Whereas this kind of loss would not be beneficial to distinguishing one document from another.

Finally, we are inspired from a query example which is “not bad hotel”. If we only consider 1-gram feature for this query, it is hard to learn the positive sentiment from the phrase “not bad”. On the contrary, the model will probably learn that “bad” is a negative sentiment word which would make the query totally opposite. How about trying n-grams (combi-

[1] http://quaid.uwaterloo.ca:33333/#/doc

nations of n sequential words) to deal with this problem? So we try classify TF-IDF values of simply 2-grams, 3-grams, combination of 1-gram and 2-grams, and combination of 1-gram, 2-grams and 3-grams separately. And we find that after combining 1-gram features with 2-grams or 3-grams, the results are much better than that of simply 1-gram feature model. Moreover, the combinations of 1-gram and 2-grams and the combinations of 1-gram, 2-grams and 3-grams almost break even for different topics. Taking computation cost into consideration, we finally only adopt the integration of 1-gram and 2-grams words as final features.

On top of using n-grams as features for document classification, query terms are also reorganized in order to compose query pairs. For example, "Deutsch Mark" is regarded as two independent terms in the baseline model. The TF-IDF valus of "Deutsche" and "Mark" are calculated separately in order to compose synthetic document for initializing train set. In our model, two terms pair are composed directly from query terms regardless of the words' relative positions. For "Deutsch Mark", we now have four candidate word pairs for composing synthetic documents which are "Deutsch", "Mark", "Deutsch Mark" and "Mark Deutsch". If the word pair doesn't appear in the vocabulary list (we only record the word whose document frequency is more than once), this pair would be removed from query pairs. In this case, "Mark Deutsch" is deleted from query pairs due to its sparsity. So the new synthetic document string would be:

$$
\begin{aligned}
&\#Rel : 1\{ \\
&\qquad Deutsch : Weight_1 \\
&\qquad Mark : Weight_2 \\
&\qquad Deutsch\ \ Mark : Weight_3 \\
&\qquad \}
\end{aligned}
$$

where $Weight_i$ corresponds to the document frequency of $i-th$ word appearing in the whole corpus. After normalization, the sum of all the $Weight_i$ would be 1 in order to make the feature vector consistent with other documents.

3.5 Query Expansion

We also use the query expansion technique to identify potential relevant documents. Given training data, that is, relevant documents and irrelevant documents labeled by human, informative terms are used to expand query and top ranked documents for a expanded query are considered as potential relevant documents to be judged. We adapt the simple mixture (SM) method [13] to expand the query. For query expansion, we want to extract informative terms from relevant documents. However, not all terms in relevant documents are informative. In SM, a background model is used to model non-informative terms.

SM assumes that terms in relevant documents are generated as below:

1. Given two models θ_0 and θ_1;
2. Given a mixing coefficient, $\overrightarrow{\pi} = (1-\pi, \pi)$;
3. For the j-th term in the i-th relevant document:
   (a) Firstly, independently generate a latent model indicator, $z_{ji} \sim \text{Bernoulli}(z|\overrightarrow{\pi})$;
   (b) Then, independently generate a term, $w_{ji} \sim d(w|\theta_{z_{ji}})$;

where π is given (eg, 0.9) , $d(\cdot)$ is a family of term distributions, θ_0 is a model for informative terms to be estimated, and θ_1 is a known background model.

In this paper, the bag-of-word assumption is used and multinomial distributions are used as term distributions, which implies $d(\cdot)$ is the family of multinomial distribution. Given a corpus and irrelevant documents obtained from human judegement, we use maximum likelihood estimation (MLE) to estimate a corpus model, θ_{corpus}, and an irrelevant model, $\theta_{\text{irrelevant}}$, respectively. The background model used in this paper is:

$$d(w|\theta_1) = 0.5 \times d(w|\theta_{\text{corpus}}) + 0.5 \times d(w|\theta_{\text{irrelevant}}) \quad (3)$$

The inference process for SM [13] is given as below:

At k-th iteration for SM,

$$\eta^{(k)}(w) = \frac{(1-\pi)d^{(k)}(w|\theta_0)}{(1-\pi)d^{(k)}(w|\theta_0) + \pi d(w|\theta_1)} \quad (4)$$

$$d^{(k+1)}(w|\theta_0) = \frac{\sum_i tf_i(w)\eta^{(k)}(w)}{\sum_{w' \in \text{voc}} \sum_i tf_i(w')\eta^{(k)}(w')} \quad (5)$$

where "voc" denotes the vocabulary for terms and $tf_i(w)$ represents the raw term frequency of w in the i-th relevant document.

Once θ_0 is estimated, we use top K ranked terms in the model to expand a query. In this paper, given a expanded query, we use the Kullback-Leibler (KL) ranking algorithm and top ranked documents are considered as potential relevant documents to be judged. The KL ranking algorithm uses the KL divergence, which measures the difference between a query q and a document d. The divergence estimates relevance of the document with respect to the query. The KL divergence is defined as:

$$KL(\theta_q||\theta_d) = \sum_w \Pr_q(w) \times [log(\Pr_q(w)) - log(\Pr_d(w))] \quad (6)$$

Note that the divergence is asymmetric. Efficiency is the main reason why an asymmetric divergence is used. Usually, a query is shorter than a document. With the help of inverted index, the divergence can be efficiently computed. On the other hand, computing a symmetric divergence is slow.

3.6 Classifier Selection

The core idea of active learning process is to make classifier continuously improved and more predictive as iterations increase. On the one hand, train set could contain more judged documents as relevance feedback increases. So it is easier to learn a continuously improved classifier with more certain labeled documents. On the other hand, due to the sparsity of relevant documents in the corpus, it would be more and more difficult to retrieve relevant documents especially during the later train phases. Is it possible to find a superior classifier to replace Logistic Regression with Pegasos updates which has been applied in the baseline model?

In order to find a superior classifier, we tried the methods shown in Table 2.

Table 2: Classifiers Applied During Experiments

Classifier	Toolbox	Feature
Logistic Regression	Sofia-ML	Unigram TF-IDF
Logistic Regression	Sofia-ML	N-gram TF-IDF
Logistic Regression	Sofia-ML	4-char TF-IDF
Linear SVM	LIBSVM	Unigram TF-IDF
Linear SVM & LR fusion	Sofia-ML	4-gram TF-IDF
RBF SVM	LIBSVM	Entropy
RBF SVM	LIBSVM	Unigram TF-IDF
Decision Tree	Scikit-Learn	Unigram TF-IDF
Naive Bayes	Scikit-Learn	Unigram TF-IDF
AdaBoost	Scikit-Learn	Unigram TF-IDF
Gradient Boosting	XGboost	Unigram TF-IDF

By recording the number of relevant documents found in each iteration, we find that the logistic regression (LR) classifier performs very well in the initial stages. The percentage of relevant documents among all the highest-scoring relevant documents returned by the classifier is around 90%. Although the train set is unbalanced in the beginning since not enough relevant documents can be found initially, the linear classifier still can efficiently dig out the relevant documents. So it would be very difficult to beat LR during the initial stages.

Derived from the above investigation and thoughts, we propose to use non-linear classifier to replace LR when the accuracy of linear classifier starts to decrease dramatically. Our first choice is the Gaussian kernel support vector machine (RBF kernel SVM), which is able to fit the maximum-margin hyperplane in a transformed high-dimensional feature space. We think that if the hyperplane can be located more precisely, the relevant documents can be found by the classifier more effectively. As for the soft margin parameter C and γ, the best combination of these two parameters would be selected by grid search with exponentially growing sequences of C and γ, for example, $C \in \{2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}\}$; $\gamma \in \{2^{-15}, 2^{-13}, ..., 2^{3}, 2^{5}\}$. Multi-fold cross validation is operated in each iteration to pick out the best combination of parameters. After applying this strategy, we find that RBF kernel SVM tends to severely overfit with large imbalanced data. Beyond that, RBF kernel SVM takes more computation and spends around 5 times training time comparing with linear classifier.

Apart from high dimensional classifier, we also tried some other classifiers, such as Stochastic Gradient Descent (SGD) linear SVM, random forest, XGBoosting and Naive Bayes classifiers. None of them can beat logistic regression. This makes sense for a random forest, which as a highly non-linear, expressive, high-variance classifier needs a relatively high ration of examples to dimensionality. Linear models are less exacting in this respect, they can even work with $d(dimensionality) \gg n(documents)$. We find only SGD linear SVM can draw with LR, so we tried the RRF fusion of two rank lists generated separately from SGD-SVM and LR. The result of this kind of fusion is still almost the same as that of baseline. In addition, cross validation over one fold of train set is also tried within these two linear classifiers, we would select the classifier with higher cross validation accuracy. However, the results is not able to improve a lot while the train time increases.

We think that a linear model is well enough for sparse high-dimensional data such as bag-of-word. If one document contains some key words or related information about a specific query, it can be regarded as relevant document. So the specific features(words or phases) related to the query terms can determine the relevance of corresponding document.

Therefore, we decide to keep using LR instead of using some fancy machine learning methods. We notice that 100 irrelevant documents in train set are selected randomly during each iteration, while this kind of randomly picking may introduce train error. Because all the randomly selected documents are not reviewed, they are labeled as “not relevant” documents presumptively. The combination of multiple LR classifiers with different randomly selected train set could be a good choice. The results show that the RRF fusion of ranking lists generated from five different LR classifiers can slightly improve the accuracy of classification. This kind of fusion might aid the classifiers in the coverage diverse aspects of the topics. Moreover it gains a lot obviously especially in the beginning stage when presumptive irrelevant documents make up a high proportion. The disadvantage of this fusion is also the high computation cost.

4. EXPERIMENTAL SET-UP

Our improved TAR process has the following phases:

1. Generate TF-IDF values for 1-gram and 2-grams features. And build index for each document in the corpus. Apply BM25 ranking to return the top 100 documents with the highest score related to a specific query.

2. Entropy is generated from TF-IDF. The entropy vector of each document is reduced to 200 dimensions by executing Latent Semantic Indexing(LSI).

3. Cluster the top 100 documents based on their LSI vectors and compose the clustering weighted graph described in section 3.1. Select documents from the most stable pairs and judge document one by one using at most 50 review efforts.

4. One synthetic document is constructed from query terms.

5. The initial relevant documents train set consists of one synthetic document and the selected judged documents from step 3. Set initial batch size B as 1.

6. Randomly select 100 documents from the corpus and temporarily label them “not relevant”. Add these presumptive not relevant documents to train set.

7. Train 5 Logistic Regression classifier with different presumptive train set. Select $\lceil \frac{4B}{5} \rceil$ documents with the highest score from fusion list for review and label them as “relevant” or “not relevant”.

8. If the prevalence of relevant document in the $\lceil \frac{4B}{5} \rceil$ documents is high, continue judging the next $\lfloor \frac{B}{5} \rfloor$ documents with the highest score.

9. Otherwise, obtain expanded terms from judged documents and generate a new ranked list based on Indri TF-IDF retrieval model ranking score. Execute RRF fusion with the list generated from step 7 and select top $\lfloor \frac{B}{5} \rfloor$ documents to review.

10. Add all the reviewed documents to the train set. Increase B by $\lceil \frac{B}{10} \rceil$. Return to step 6 and start the next iteration until all the documents in the corpus have been reviewed.

Following the baseline [3], our implementation used a feature space consisting of words (1-gram and 2-grams) occurring at least twice in the collection, and, following [8], Porter stemming, elimination of SMART stopwords, and Cornell ltc term weighting. Indri 5.9 provides BM25 and TF-IDF retrieval model methods to retrieve documents. So we select Indri to build index and rank documents during seed selection and query expansion phases.

LSI operation involves dimensionality reduction and singular value decomposition (SVD) which requires numerous calculations. RedSVD is an effective tool to accelerate SVD computation and can shorten the generation time around 3 times. As for clustering, we use K-Means clustering method from Scikit Learn package [9]. Based on n top ranked documents with the highest BM25 score, we set $\log n$ as k clusters. In this case, k is 7 where $n = 100$.

For the learning algorithm, we still choose the Sofia-ML implementation of Pegasos SVM, with the following parameters: "–iterations 2000000 –dimensionality 110000". However, the dimensionality needs to be dynamically updated according to the size of features. For large corpus and n-grams features, we may increase the dimensionality.

The Total Recall organizers provide three different dataset modes (trivial, test and bigtest) for testing, which respectively contain 7 topics with 2 datasets containing 30 documents each, 7 topics with 2 datasets (Oldreut and 20ng) containing about 20,000 documents each and 2 topics with one dataset (Enron) containing about 750,000 documents. The corpora are mostly made by news articles and emails. In order to verify the effectiveness of our model, we also test our method on 50 selected topics with high prevalence of relevant documents from Newreut and Robust04 corpora.

There are two tasks to submit the final results. One is Play-at-Home in which we can run our own systems locally and access the automated assessors via the Internet. The other one is SandBox in which we should set up our systems in the virtual machine. And we submit the virtual machine so that the coordinators will execute our system within a restricted environment. The methods of two submissions are almost the same. However, there are some slight differences between these two deployments, for example, the configuration for virtual machine environment. We upload our Play-at-Home code to bitbucket[2].

5. PRIMARY RESULTS

In order to evaluate our models, we run our systems on different types of corpus, such as Oldreut, 20ng, Enron, Newreut and Robust04 respectively. We only list the results from Oldreut and 20ng in this paper. Because most of the documents in these two corpora have been judged so the results from them are more reliable and persuasive. Whereas for Newruet and Robust04, both of them only have a small portion of documents judged and the prevalence of relevant documents is also small compared to Oldreut and 20ng.

Table 3: 75% Recall Effort of Different Methods

Topic	SD_2gram	SD_3gram	SD_2gram_QE	Baseline
tr0	2119	2150	**2107**	2145
tr1	**118**	183	137	164
tr2	**157**	243	188	273
tr3	215	**208**	247	248
tr4	762	763	762	**761**
tr5	761	762	762	**760**
tr6	764	**761**	774	765

In general, there are two standard ways to evaluate the results: as gain curves and as 75% recall-effort values. A gain curve keeps track of the number of efforts to reach a certain number of relevant documents. Seven topics from oldreut and 20ng are listed from Figure 1 to Figure 7. As shown in Table 3, the 75% recall-effort values records the amount of efforts to achieve $recall = 0.75$. The gain curve enables us more intuitively to look at the effort spent for any given level of recall. From the figures and table, we can find that our improved models can achieve 75% recall with less effort for topic: tr0 to tr3. As for the topics from tr4 to tr6, there is no clear winner among the improved models and baseline. All the methods are close and the accuracy of baseline are nearly 92% and our models still performs slightly better than the baseline.

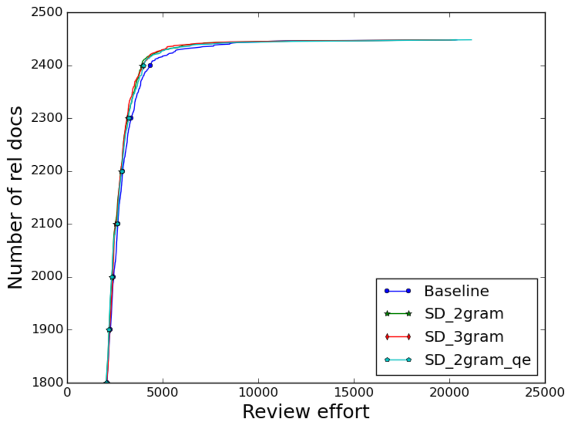

Figure 1: tr0-Relevant Documents vs. Review Effort

6. CONCLUSIONS

Our experiments show that clustering based seed selection, extended TF-IDF which includes 2-gram and 3-gram features and continuous query expansion, in combination will

[2] https://bitbucket.org/HaotianZHANG/uwtotalrecall

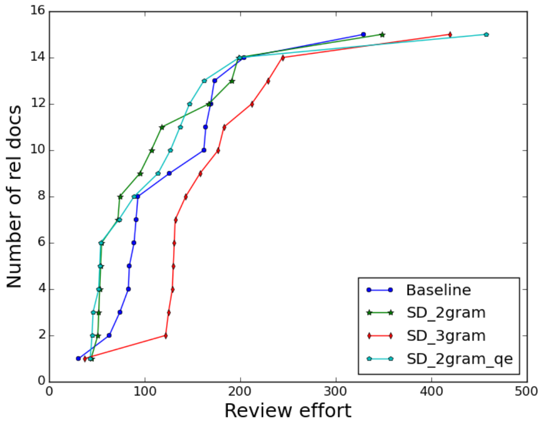

Figure 2: tr1-Relevant Documents vs. Review Effort

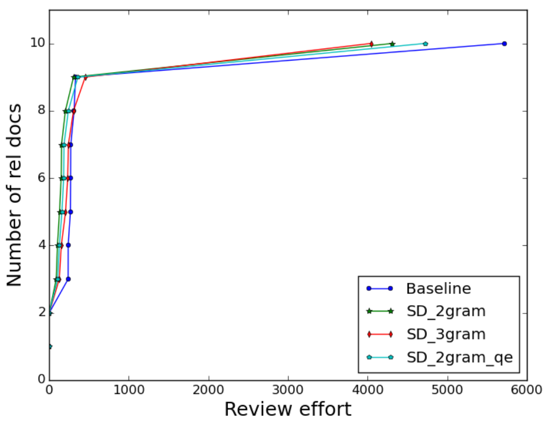

Figure 3: tr2-Relevant Documents vs. Review Effort

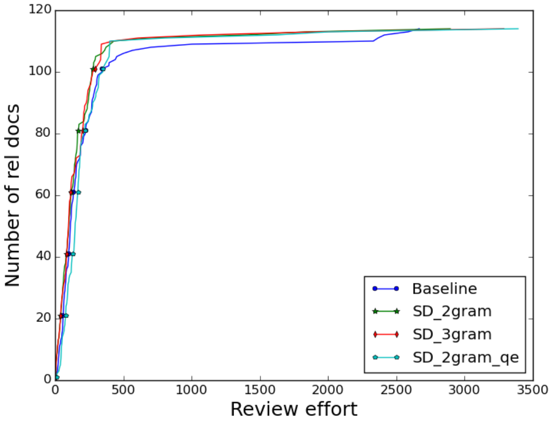

Figure 4: tr3-Relevant Documents vs. Review Effort

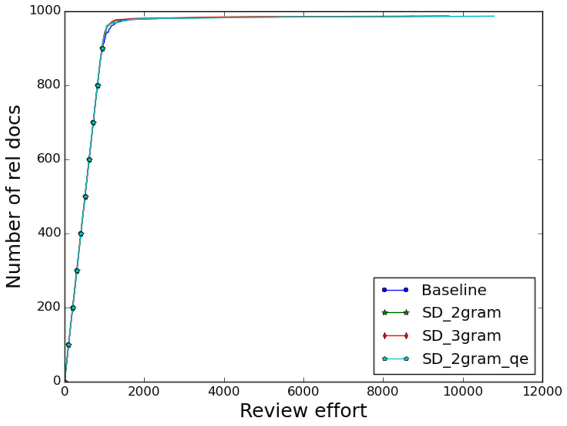

Figure 5: tr4-Relevant Documents vs. Review Effort

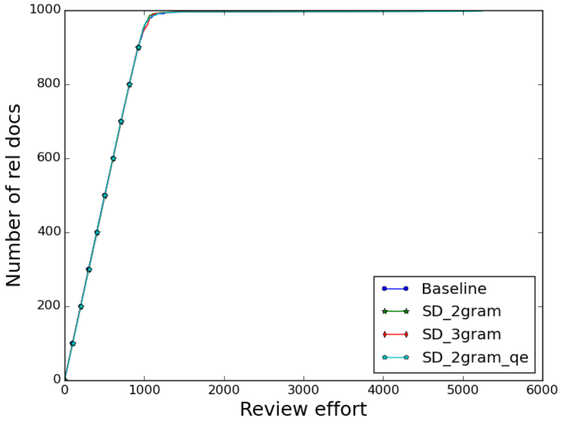

Figure 6: tr5-Relevant Documents vs. Review Effort

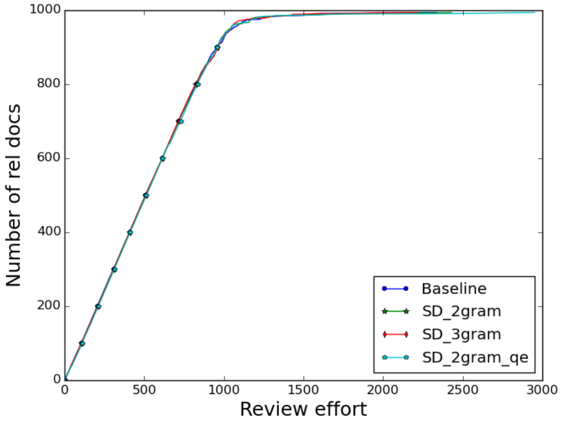

Figure 7: tr6-Relevant Documents vs. Review Effort

achieve high recall with less effort. As for classifier selection, the linear classifier performs well enough for sparse high-dimensional data. We also find that the performance of our models varies for different kinds of topics and datasets. How to wisely choose strategies to deal with different situations remains a problem.

7. ACKNOWLEDGMENTS

We thank Prof. Gordon Cormack and Adam Roegiest for their helpful guidances and feedback. We also would like to thank Prof. Yi Zhang and Prof. Maarten de Rijke for their useful advice during ACL2015.

8. REFERENCES

[1] G. V. Cormack, C. L. Clarke, and S. Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759. ACM, 2009.

[2] G. V. Cormack and M. R. Grossman. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 153–162. ACM, 2014.

[3] G. V. Cormack and M. R. Grossman. Autonomy and reliability of continuous active learning for technology-assisted review. *arXiv preprint arXiv:1504.06868*, 2015.

[4] G. V. Cormack and M. Mojdeh. Machine learning for information retrieval: Trec 2009 web, relevance feedback and legal tracks. In *TREC*, 2009.

[5] J. P. Higgins, S. Green, et al. *Cochrane handbook for systematic reviews of interventions*, volume 5. Wiley Online Library, 2008.

[6] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[7] C. Hogan, J. Reinhart, D. Brassil, M. Gerber, S. M. Rugani, and T. Jade. H5 at trec 2008 legal interactive: user modeling, assessment & measurement. Technical report, DTIC Document, 2008.

[8] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 33–40. ACM, 2004.

[11] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[12] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[13] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410. ACM, 2001.

[14] L. Zhang and Y. Zhang. Interactive retrieval based on faceted feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 363–370. ACM, 2010.

[15] L. Zhang, Y. Zhang, J. de Arma, and K. Yu. Ucsc at relevance feedback track. Technical report, DTIC Document, 2009.