11-2014

Towards a Simple and Efficient Web Search Framework

Di Xu
Carnegie Mellon University

Jamie Callan
Carnegie Mellon University, callan@cs.cmu.edu

Published In
The Twenty-Third Text REtrieval Conference (TREC 2014) Proceedings, SP 500-308.

Towards a Simple and Efficient Web Search Framework

Di Xu, Jamie Callan

Language Technologies Institute, School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA

dix@cs.cmu.edu, callan@cs.cmu.edu

Abstract

The Web Track of 2014 Text REtrieval Conference (TREC) addresses the most fundamental problem of Information Retrieval. We did not intend to craft a system that beats the state-of-the-art search engines, but to design a light weight and cost-effective system with comparable performances. We introduce a two-pass retrieval framework, with the first pass consisting of a simple and efficient retrieval model that focuses on recall, and the second pass a wave of feature extraction algorithms run on the set of top ranked documents, followed by Learning to Rank (LETOR) algorithms that provide different precision oriented rankings, and their outputs are combined using data fusion. We have focused on using statistical Language Models with novel and well-known smoothing techniques, different LETOR methods, and various data fusion techniques. In addition, we have also tried using topic modelling with Hierarchical Dirichlet Allocation for query expansion in the hope of improving diversity of our results. However, the topic modelling approach has turned out to be unsuccessful, and we have not been able to spot the problem and benefit from it in this work. In addition, we also present some further analyses demonstrating that our approach is robust against overfitting, and some general studies on overfitting in the context of LETOR.

Index Terms: Information retrieval, search engine, language Model, learning to rank, machine learning, data fusion

1. Introduction

Ad-hoc retrieval addresses the most fundamental problem of Information Retrieval (IR) and has been studied extensively for decades. Cliche as the topic sounds to most IR researchers, there has been no single retrieval model proposed in the past that consistently out performs others in general, and none of the existing approaches has the performance that is close to human. In essence, the task requires the machine to be equipped with the ability to accurately rank a document among others according to its relevance to a given query, which requires a high level of algorithmic approximation to human intelligence. The algorithm not only needs to be able to understand the content of a document and compare it with other candidates, but also to interpret a query and enumerate multiple topics that the query may address. Therefore, as long as Artificial Intelligence remains under study, the fundamental problem of IR will continue to remain unsolved.

The introduction of the well-known retrieval models introduced in the past decades can be found in many well written literatures such as [1], thus we will skip the introduction of those well-known retrieval methods such as BM25, Vector Space Models and Language Modelling approaches. In short, the methods that have proved to be effective eventually come to an agreement when calculating the statistics from documents. It is remarkable that tf-idf heuristics still lies in the hard of most retrieval methods, although they may be developed upon different theories and assumptions.

Various applications of Machine Learning (ML) methods in IR tasks has been proposed and studied extensively in the past decade. It is becoming even more popular nowadays with the advancement of ML techniques, bringing IR up to a new level. The typical form of applying ML techniques in retrieval tasks is Learning to Rank (LETOR). One advantage of LETOR is that it allows incorporating features that addresses various characteristics of a document and its relevance to a topic. Many of such features cannot be easily integrated in the formulae of conventional retrieval models due to lack of theoretical foundations. Therefore, leaving them to a regression model can at least harvest some benefits, even though we do not know the exactly correct way of calculating the relevance scores.

In spite of the abundant works done in the past, most works were validated on relatively small datasets and failed to address the scalability issue in the scale of the web. Web scale applications impose challenges upon the original ad-hoc retrieval problem, as the number of documents to be ranked is magnitudes larger for some existing system to handle. This is particularly important as web services are meant to be fast and reliable. Another important aspect of web search is that a significant portion of web texts are spam that seriously affects the retrieval quality, while most retrieval models are quite vulnerable to spam. Therefore, the precision of web search is typically low. On the other hand, diversity has also been regarded as an important aspect of web search quality, because many top ranked documents tend to suffer from redundancy issues.

People participated in previous years' Web Tracks have tried improving retrieval performances from many aspects. Some tried enhancing the low-level knowledge representation of text, such as using Latent Semantic Indexing. More recently, in [2], Quantum Language Models were used, and documents and queries were associated with a density matrix. Some other works such as [3], where clustering was performed on the documents to exploit document wise distance and semantic relationships to improve document ranking. To tackle with noisy web text, [4] tried utilizing the name entities from the Freebase Dump provided by Google Inc., which was reported to be effective in enhancing precision. Moreover, in their work, term proximity was also considered beyond the bag-of-words representation, and a modified version of BM25 was used to incorporate phrase frequencies when computing document scores. Apart from those, there are many other promising approaches presented in the past. However, very few of them are promising in a practical sense, because most of them rely heavily on sophisticated document representations and machine learning algorithms, which are not scalable in real world web search applications.

We aim to design a simple precision oriented system that does not rely on external resources such as spam ranking scores or introduce extra processing engines such as name entity recognizer. We expect our system to be able to retrieve documents as efficient as possible, without sacrificing too much of the performance and still achieve competent precision and normalized Discounted Cumulative Gain (nDCG).

In the rest of this paper, we will introduce the general pipeline of our retrieval system in Section 2, followed by an introduction to novel Language Modelling approaches in Section 3 that were used to generate features, and later in Section 4 we introduce all other features and the LETOR algorithms. In addition, we also introduce our failed attempt of using topic models to perform query expansion to capture documents of multiple topics in section 5, and a brief reflection on why this approach did not work. Finally in section 7 we report and discuss the evaluation results.

2. General Pipeline

Our goal is set to design a system as simple as possible, without using any external processing engine or resources, other than the standard Indri toolkit and a third party LETOR toolkit. We have implemented most of our ranking algorithms implemented using Lucene.

We introduce a two-pass retrieval framework, where in the first pass we aim to retrieve as many relevant document as possible to ensure a reasonable level of recall, and in the second pass we process all the retrieved documents in the first pass and extract features. Those features are then piped into different LETOR algorithms to produce several rank lists, and eventually all the rank lists are merged using the conventional Reciprocal Rank based data fusion method.

In detail, in the first pass we use the standard Indri retrieval algorithm and BM25 with pseudo relevance feedback on the top 10 highest ranked documents. The parameters of the two-stage smoothing used by Indri and BM25 were tuned to optimize recall instead of precision, since the goal in the first pass is to secure the recall of the final ranking. The goal of using both Language Model based two-stage smoothing and tf-idf based BM25 is that although they have demonstrated the same level of performance empirically, the two methods are complimentary as they tend to retrieve different sets of relevant documents. Therefore by merging the results returned by both rankers, we can harvest more relevant documents for further processing and re-ranking. The implementation of Indri's Search Engine allows fast and parallel search over the entire ClueWeb12 corpus. After we obtain the two rank lists generated using Indri, we fuse them using the Reciprocal Rank method to generate a baseline rank list, which is the final product of the first pass. In the fused rank list, we reserve only the top 100,000 documents for each query.

In the second pass, we extract various features that address the relevance of a document to the query, and also some other features that independently reflects the document's characteristics. In particular, the features we have extracted consists of document-level and field-level relevance scores computed using some well-known and novel ranking methods, and some simple heuristic statistics that are likely to be associated to spam, and other basic features such as document length. We have also studied some less recognized but interesting language models in Section 3. A more detailed summary of all features used in this stage and the ensuing LETOR algorithms are presented in Section 4. Multiple LETOR algorithms were used in this stage to provide different rank lists for the same query. To properly merge these results, we have compared several data fusion techniques, including Reciprocal Rank, Borda Count and Condorcet method, and have found that Reciprocal Rank is more effective than the other two methods. A brief introduction and comparison of the three methods will be discussed in Section 6.

3. Discriminative Language Models

There have not been many efforts invested recently in the development of statistical language models (LM) and smoothing techniques applied in Information Retrieval. One important reason is that the assumption on the distribution of token types that most LMs are based does not hold noisy data such as web text or twitter data. In web text, the type-token curve is less linear comparing to those of standard text corpus such as Wall Street Journal where most LMs and smoothing techniques were validated. LMs need to be able to evaluate query terms in a discriminative manner in order to do better in web search.

One simple idea is to introduce a Negative Language Model (NLM) that accounts for common terms that are less meaningful and mostly useless. This is analogous to idf heuristics, but has better statistical foundations in the domain of statistical LMs. One interesting work on NLM that we consider is "Negative Query Generation" proposed by Zhai and Lv in [5]. In their work, a denominator is introduced which addressed the generation probability of a "negative query" from a document, which they interpret as the probability that a user who dislikes the document would choose to use this query. Intuitively, this can be regarded as measuring how far the query is from the document in terms of the distance between their corresponding LMs. With this idea, they have extended the traditional Dirichlet Smoothing to discriminate documents based on the corresponding query likelihood with negative query generation. Their reported results on WT10G have demonstrated a certain level of success. Therefore, we have implemented their proposed LM with negative query generation, denoted as Dir-XQL.

Motivated by XQL, we have also come up with a similar discriminative model based on Jelinek-Mercer Smoothing, denoted by JM-XQL.

4. Features and Learning to Rank

Apart from Indri, BM25, Dir-XQL and JM-XQL, we have also implemented some other smoothing techniques such as Good Turing and Absolute Discounting, and different similarity measures such as cosine similarity and KL divergence. In addition to the scores of the document given by different rankers, we have also computed the scores particularly for the body and anchor text of the documents, as they are likely to contain information about the topic if the document is relevant. Moreover, we have also recorded the corresponding scores for each query term, and on top of that, computed the harmonic mean, geometric mean, variance and skewness. Our assumption is that if a document is relevant to the query, it should address as many of the query terms as possible.

We have also designed some heuristics features. We have introduced a binary value addressing if at least one of the query terms appeared in the title and URL, as we assume some of the relevant documents may have parts of the query in their title and URL. Moreover, the contextual distribution of the query terms are investigated and for terms that appear more than once in a document, we measure the mean, variance and skewness of the contextual distance between their occurrences, normalized

by the length of the document. These features are designed to target some query terms that appear too frequently, thus are less meaningful and more likely to associate with spam.

Nonetheless, we also included some basic document statistics such as the document vocabulary size, field lengths, and skewness of document term frequencies. These features are helpful in lowering the score for very long and potentially spam documents. There are also features taken from the query that are independent from documents, including query length, the average, minimum, maximum of the collection frequencies of the query terms.

Multiple LETOR methods have been tried, which are different in many ways and we expect them to be complimentary during the final fusion.

Simple K-nearest neighbour (KNN) with K set to 20 and Regression Tree was used to perform point-wise LETOR. They are expected to works well when the features independently address different aspects of the documents, but are more sensitive to noises and less effective when the dimension of the feature vector is too high. Our intuition is that rank lists generated by point-wise methods are better at the top potions, but the precision drops quickly if we go further down the list, as they are prone to over-fit to certain features that are most dominating.

We have tried using Support Vector Regression (RankSVM) with linear kernel for pairwise LETOR, and were trained on a set of error pairs collected using the "web2013" relevance judgments file. We expect the pairwise methods to perform better than point-wise approaches, as the features collected from the error pairs are more meaningful as they define relative distances.

For list-wise LETOR, we are using ListNet [6], which uses a simple one layer Neural Network with Gradient Decent to optimize a defined list-wise loss function based on "top one probability". The list-wise approach was proved to be more effective than pairwise and point-wise approaches, as its optimization criterion is closer to the actual evaluation metrics. According to Cao et al. (2007), the loss function of the pairwise LETOR methods are not inversely correlated with NDCG or other precision oriented metrics, whereas the loss function for list-wise LETOR methods was demonstrated to be completely inversely correlated to those metrics. They also pointed out that pairwise methods converge more slowly than list-wise methods, making it more difficult to train and reach an optimal solution. For both RankSVM and ListNet, we have adopted the implementations from RankLib, which is part of the Lemur Project [7].

In addition, we have also tried using Genetic Programming (GP) [8] based LETOR, which is a new generation of LETOR approaches. The implementation of the GP learner was provided by "the Learning to Rank Library" from Yandex School of Data Analysis [9]. According to the authors, GP based LETOR was able to achieve competitive performance with RankSVM and RankBoost, but its computational cost is higher. The nature of GP algorithms is also prone to overfitting.

All of the eager learning models were trained with 10-fold cross validation.

It is also worth noticing that even though most of these features are directly consistent to the relevance of a document to a query, none of our LETOR methods include diversity into account. This is because we were counting on topic modelling based query expansion to improve diversity performance, such that we have not defined a dedicated list-wise optimization criterion on top of the rank list that addresses diversity. How to optimize towards diversity under the context LETOR is yet another problem to be studied in future.

5. Query Expansion with Topic Modelling

Topic models have been widely studied for a long time [10] and has proved to be useful in many applications [11], even for applications that does not deal with natural languages at all [12]. Latent Dirichlet Allocation (LDA) [13] is one popular implementation of topic models and have demonstrate its effectiveness in many tasks [14] [15].

We have tried query expansion based on topic models to address the diversity issue which is natural to web search. The intuition is to use LDA to identify potential topics from the top ranked documents. Originally, this was performed after the first pass when most of the relevant documents are assumed to be retrieved by using BM25 and Indri with pseudo relevance feedback. And we hopped to be able to generate weighted distribution of words that could potentially identify multiple topics of a query from the top ranked documents, and by using these approximations of multiples topics, we can perform multiple searches for the same query with different expansions, followed by separate LETOR for each expanded query, and eventually merge the results with data fusion.

Unfortunately, the LDA based topic mining approach has failed in this task. The resulting topics generated by the topic model did not carry any useful information about the various aspects of a topic. For example, for the query "raspberry pi", it covers topics such as "what is raspberry pi", "making a raspberry pi". However, the topics generated based on the 10 top ranked documents do not make much sense to us in terms of their keywords, as presented in Table 1. It is obvious that the "topics" generated by LDA do not really characterize the real topics of relevance, and were completely overwhelmed by words that have nothing to do with the query.

Topic	Key terms in topic
1	*blog,fedora,boards,35,manufacture,march,price*
2	*jacob,colours,cut,acrylic,related,blogthi,twitter*
3	*hardware,high,finally,propaganda,suede,batch,vodka*

Table 1: Top terms (with the highest weights) for the topics generated by the LDA topic model for query "raspberry pi".

One simple explanation is that web texts are too noisy and unfocused for the LDA process to stabilize on the real topics that we are interested. This makes sense because most web documents does not focus on one specific topic, the vocabulary is large, and thus the LDA requires more documents of the same topic to take effect. However, under the context of web search, this is not feasible because very few documents are as focused as what the LDA model expects. In order for the topic modelling approach to work as we expect, the first pass rank list must already have a high precision in the top of the list, but this is not feasible in the first pass as our first pass retrieval focuses on recall instead of precision. Therefore, we can conclude that in the context of web search, we cannot use topic modelling approach to extract topics in an early stage.

One potential alternative is to use the Dominating Set Approximation method (DSPapprox) [16] on the top portion of the first pass rank list iteratively, which is yet another problem to be studied in future.

6. Data Fusion

Data fusion has been proved useful in improving retrieval performances, especially when the systems to be combined carry

different sources of information and are complimentary to each other. Data fusion algorithms can be divided into two groups, with one utilizes the scores of the posting lists during the combination, while the other considers only the rank positions. Many previous studies on Data fusion [17] [18] [19] suggested that when the scores of the systems to be combined are commensurable, using score based fusion methods are better than using only the rank positions, but when the scores are incompatible or if the systems generate different rank-score curves, rank based fusion techniques are better. The latter is typical in our case because the scores generate by different LETOR algorithms are different in terms of scale and rank-score curves.

In particular, we have compared Reciprocal Rank, Borda Count [20] and Condorcet method [21]. The latter two methods came from social theory of voting. On the Web Track 2013 query set, we performed data fusion on the posting lists generated by some of the LETOR algorithms mentioned above. We chose to only use the top ranked 100 documents to perform the experiments because Condorcet method requires global ranking information and does not scale with much longer posting lists.

We have observed that Reciprocal Rank significantly outperformed Borda Count and Condorcet method by more than 0.03 absolute in prec@30 and more than 0.05 in nDCG@30, whereas the performance of the latter two were very similar. This observation is similar to that in [22], and it is likely to be the case that false positives that are common in all 4 posting lists will likely to receive higher ranking than true positives that are supported by a subset of posting lists. Our LETOR algorithms behave differently on some topics, but Condorcet method tends to ignore high votes from the minority, but instead prefer weak votes from the majority. Therefore, we have adopted Reciprocal Rank as the data fusion techniques in our final submissions.

7. Evaluation Results and Analyses

7.1. Submission Results

The evaluation results for our submissions on Web Track 2014 are shown in Table 2 and 3. We have submitted 3 runs. *Zerg* run came from a primitive system that does not perform LETOR, and instead performs data fusion directly on posting lists generated by different rankers such as BM25, Indri, XQL, etc., which we regard as our baseline performance since LETOR was not applied. *Protoss* run was generated exactly by our two-pass retrieval and ranking system introduced in this paper. *Terran* was a slight modification of *Protoss*, and it did not involve KNN during the data fusion stage. We excluded KNN for the consideration because it is a lazy learning algorithm which contradicts to our goal of generating a simple and efficient retrieval framework, and also for the fact that KNN is unstable and sensitive to noises.

	ERR@20	p	nDCG@20	p
median	0.1667	-	0.2548	-
Zerg	0.1740	0.2667	0.2670	0.1442
Protoss	0.1968	0.0098	0.2864	0.0092
Terran	**0.2043**	0.0090	**0.2940**	0.0064

Table 2: Results for standard gdeval metrics. *P*-values were computed using directional Paired t-Test against Median.

It can be observed from Table 2 that our systems are generally better than the median in terms of gdeval metrics where diversity is ignored, especially for *Protoss* and *Terran*, which are shown to be significantly better with strong evidence. This also suggests that our LETOR framework is effective in improving the overall precision.

	ERR-IA@20	p	P-IA@20	p	α_nDGC@20	p
median	0.5747	-	0.4364	-	**0.6592**	-
Zerg	0.5360	.06	0.4364	.50	0.6289	.06
Protoss	0.5693	.42	0.4461	.24	0.6398	.19
Terran	**0.5779**	.45	**0.4529**	.12	0.6467	.27

Table 3: Results for standard ndeval (diversity) metrics. *P*-values were computed using directional Paired t-Test against Median.

We are not surprised that our systems did not work well on diversity metrics as shown in Table 3, because the diversity module of our system was not functioning as we expected and eventually we chose to not to include it in our pipeline. Still, at least we are not significantly worse than the median.

7.2. Overfitting Analyses

We are interested in whether our features are effective and to see if the LETOR models trained on top of those features are robust against overfitting. We have conducted several sensitivity analyses on the learning curve of ListNet on both the training (Web 2013) and testing (Web 2014) query sets. Our assumption is that if our features are effective and not random there is less chance for the model to overfit to randomness in the training data, thus its performance on the training query set is supposed to be close to that on the testing query set.

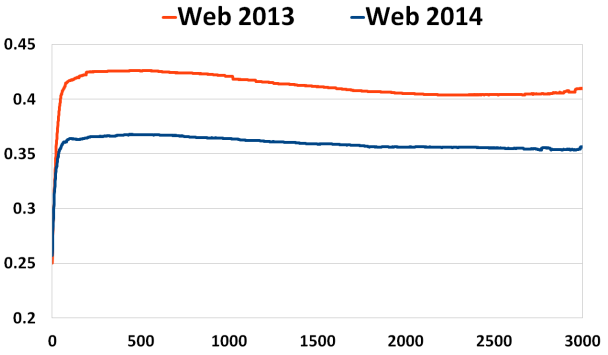

Figure 1: The learning curve of ListNet in the first 3000 iterations, with learning rate set to 0.01. The X-axis is the number of iterations for the training of the 1-layer neural network (NN), the Y-axis is the optimization criterion correlated to MAP. The features were collected based on the full ClueWeb12 dataset.

It can be observed through Figure 1 that our features work well with ListNet in terms of robustness against overfitting. This experiment indicates that our features are not sensitive to the queries because we are validating on a different query set. This suggests that our features are stable with respect to different kinds of queries.

Following this intuition, we are also interested in whether our features are still robust when the underlying dataset has changed. To simulate such condition, we took one step further and re-performed the feature extractions on the ClueWeb12-B13 dataset, which is a small sample (50 million documents) of the full dataset (733 million documents). We then observed

again how the LETOR performance differs on the two query sets.

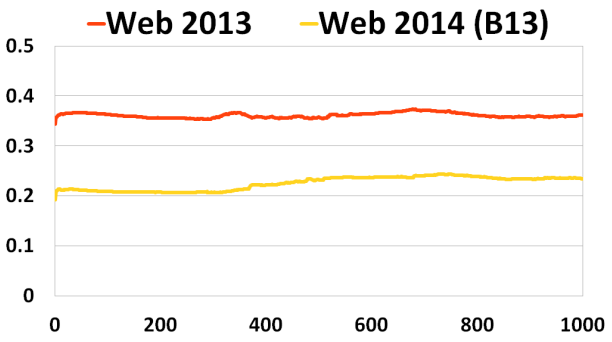

Figure 2: The learning curve of ListNet in the first 1000 iterations, with learning rate set to 0.1. The X-axis and Y-axis are the same as in Figure 1. The features for Web 2013 were collected based on the full ClueWeb12 dataset, but those for Web 2014 are collected on ClueWeb-B13 dataset.

According to Figure 2, it appears that the performance of ListNet on Web 2013 and 2014 are still consistent with respect to the number of iterations used for the NN training, even if for Web 2014 the features were extracted based on a much smaller dataset. This also suggests that our features are not sensitive to the change of dataset, and they reflect general characteristics of a document being relevant or irrelevant to a general query.

7.3. Reflections

In table 4, we compare our gdeval results with some well-known teams who have participated in the Web Track in the past. It is remarkable that top ranked teams such as ICTNET udel_fang and uogTr all featured in using the provided Freebase entity annotations to achieve impressive performances.

Group	Run	ERR@20	nDCG@20
ICTNET	ICTNET14ADR1	0.208 (1^{st})	0.261 (6^{th})
udel_fang	UDInfoWebAX	0.207 (2^{nd})	0.307 (2^{nd})
Group.Xu	Terran	0.204 (3^{rd})	0.294 (3^{rd})
uogTr	uogTrDwl	0.195 (4^{th})	0.324 (1^{st})
UMASS_CIIR	CiirAll1	0.153 (11^{th})	0.250 (10^{th})

Table 4: This table documents the officially released evaluation results for our (Group.Xu) submission *Terran* and the best submission from some of the "well-known" participants in the Web Track. The number inside the parenthesis after each fraction is their ranking among all the participants.

Comparing to all automatic runs, our *Terran* run won the 3rd place in the list in terms of both ERR@20 and nDCG@20, according to **Table 2** of the TREC 2014 Web Track Overview [24]. This suggests that our approach works reasonably well and our goal of constructing an low-cost and effective retrieval framework was a partial success, in terms of non-diversity metrics. We have chosen not to use the provided Freebase entity annotations because it contradicts to our goal of keeping the system simple, because entity annotations are not always available in real world and real time web search.

Of course, our goal has not been completely fulfilled yet because our diversity performance was mediocre at best. We were not surprised because we eventually gave up on improving the diversity performance, as we also need to submit our result for the Contextual Suggestion Track. It is worth noticing that the teams that achieved good diversity performance all featured in using entity based query expansion techniques. Therefore, improving diversity without explicit entity recognition will be our challenge in future.

8. Conclusions

We can conclude that our precision oriented system works well in generating ranked lists with competitive precision, and is much simpler comparing to many more sophisticated systems in the pass since it does not require any extra resources or external tool-kits. Our designed features and the LETOR framework have achieved a level of success in dealing with spam texts and improving the overall ranking quality, and we have demonstrated the effectiveness of our features which incur limited effects of overfitting when learned with ListNet. We regret that we could not get our LDA based topic model to work in mining different themes of a query. In future, we plan to introduce simpler and more effective strategies to improve the diversity performance, such as DSPApprox [16] and maximum entropy methods. We would also explore new document level features to make the ranking system less sensitive to spam documents.

9. Acknowledgements

This work was supported in part by the Language Technologies Institute of Carnegie Mellon University. We would like to thank David Pane for providing the ClueWeb12 datasets and the Indri index for both ClueWeb12 and ClueWeb12-B13.

10. References

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[2] A. Sordoni, W. Yuan, and J.-Y. Nie, "Universit de montral at trec 2013: Experiments with quantum language models in the web track," DIRO, Universit de Montral, Tech. Rep., 2013.

[3] F. Raiber and O. Kurland, "The technion at trec 2013 web track: Cluster-based document retrieval," Faculty of Industrial Engineering and Management Technion Israel Institute of Technology, Tech. Rep., 2013.

[4] Y. Xue, F. Guan, X. Yu, Y. Liu, and X. Cheng, "Ictnet at web track 201," Chinese Academy of Sciences and University of Chinese Academy of Sciences, Tech. Rep., 2013.

[5] Y. Lv and C. Zhai, "Query likelihood with negative query generation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1799–1803.

[6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 129–136.

[7] J. Allan, J. Callan, K. Collins-Thompson, B. Croft, F. Feng, D. Fisher, J. Lafferty, L. Larkey, T. N. Truong, P. Ogilvie *et al.*, "The lemur toolkit for language modeling and information retrieval," *The Lemur Project.[WWW document] http://lemurproject. org (accessed 25 January 2012)*, 2003.

[8] J.-Y. Yeh, J.-Y. Lin, H.-R. Ke, and W.-P. Yang, "Learning to rank for information retrieval using genetic programming," in *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007)*, 2007.

[9] Y. S. of Data Analysis, "Learning to rank library," 2013.

[10] D. Hall, D. Jurafsky, and C. D. Manning, "Studying the history of ideas using topic models," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2008, pp. 363–371.

[11] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li, "Comparing twitter and traditional media using topic models," in *Advances in Information Retrieval*. Springer, 2011, pp. 338–349.

[12] M. Bicego, P. Lovato, B. Oliboni, and A. Perina, "Expression microarray classification using topic models," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 1516–1520.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[14] 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*. ACM, 2001, pp. 403–410.

[15] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining eclipse developer contributions via author-topic models," in *Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on*. IEEE, 2007, pp. 30–30.

[16] D. Lawrie, W. B. Croft, and A. Rosenberg, "Finding topic words for hierarchical summarization," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 349–357.

[17] D. F. Hsu and I. Taksa, "Comparing rank and score combination methods for data fusion in information retrieval," *Information Retrieval*, vol. 8, no. 3, pp. 449–480, 2005.

[18] R. Nuray and F. Can, "Automatic ranking of information retrieval systems using data fusion," *Information Processing & Management*, vol. 42, no. 3, pp. 595–614, 2006.

[19] K. Mc Donald and A. F. Smeaton, "A comparison of score, rank and probability-based fusion methods for video shot retrieval," in *Image and video retrieval*. Springer, 2005, pp. 61–70.

[20] D. Black, "Partial justification of the borda count," *Public Choice*, vol. 28, no. 1, pp. 1–15, 1976.

[21] M. Montague and J. A. Aslam, "Condorcet fusion for improved retrieval," in *Proceedings of the eleventh international conference on Information and knowledge management*. ACM, 2002, pp. 538–548.

[22] 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*. ACM, 2009, pp. 758–759.

[23] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, pp. 23–581, 2010.

[24] K. Collins-Thompson, P. Bennett, F. Diaz, C. L. Clarke, and E. M. Voorhees, "Trec 2014 web track overview," MICHIGAN UNIV ANN ARBOR, Tech. Rep., 2014.