Supervised Approaches for Explicit Search Result Diversification

Sevgi Yigit-Sert[a], Ismail Sengor Altingovde[a,*], Craig Macdonald[b], Iadh Ounis[b], Özgür Ulusoy[c]

[a] *Computer Engineering Department, Middle East Technical University, Ankara, Turkey*
[b] *School of Computing Science, University of Glasgow, Glasgow, UK*
[c] *Computer Engineering Department, Bilkent University, Ankara, Turkey*

Abstract

Diversification of web search results aims to promote documents with diverse content (i.e., covering different *aspects* of a query) to the top-ranked positions, to satisfy more users, enhance fairness and reduce bias. In this work, we focus on the explicit diversification methods, which assume that the query aspects are known at the diversification time, and leverage supervised learning methods to improve their performance in three different frameworks with different features and goals. First, in the LTRDiv framework, we focus on applying typical learning to rank (LTR) algorithms to obtain a ranking where each top-ranked document covers as many aspects as possible. We argue that such rankings optimize various diversification metrics (under certain assumptions), and hence, are likely to achieve diversity in practice. Second, in the AspectRanker framework, we apply LTR for ranking the aspects of a query with the goal of more accurately setting the aspect importance values for diversification. As features, we exploit several pre- and post-retrieval query performance predictors (QPPs) to estimate how well a given aspect is covered among the candidate documents. Finally, in the LmDiv framework, we cast the diversification problem into an alternative fusion task, namely, the supervised merging of rankings per query aspect. We again use QPPs computed over the candidate set for each aspect, and optimize an objective function that is tailored for the diversification goal. We conduct thorough comparative experiments using both the basic systems (based on the well-known BM25 matching function) and the best-performing systems (with more sophisticated retrieval methods) from previous TREC campaigns. Our findings reveal that the proposed frameworks, especially AspectRanker and LmDiv, outperform both non-diversified rankings and two strong diversification baselines (i.e., xQuAD and its variant) in terms of various effectiveness metrics.

Keywords: explicit diversification, supervised learning, query performance predictors, aspect importance.

*Corresponding author
Email addresses: sevgi.yigit@metu.edu.tr (Sevgi Yigit-Sert), altingovde@ceng.metu.edu.tr (Ismail Sengor Altingovde), Craig.Macdonald@glasgow.ac.uk (Craig Macdonald), Iadh.Ounis@glasgow.ac.uk (Iadh Ounis), oulusoy@cs.bilkent.edu.tr (Özgür Ulusoy)

1. Introduction

Diversification is an approach to satisfy the needs of a population of users for ambiguous and/or broad queries, by ensuring that the documents addressing different possible intents of users are surfaced to the top results. Consider the ambiguous query "apple" – the top-ranked documents should cover both possible *aspects* (a.k.a., subtopics or interpretations) of this query, namely, apple as a fruit and the company. For a broad query, say, "machine learning", there is a wide range of aspects, such as the technological aspects (e.g., learning algorithms, code repositories), the social aspects (e.g. ethics, jobs), or the legal aspects (e.g. bias, fairness), which should all be represented – as much as possible – in an unbiased result set. Therefore, given an initial retrieval result for a query, usually called a *candidate set*, diversification methods are applied to generate a final ranking, which lists top-ranked documents that are both relevant and diverse, i.e. covering as many aspects of the query as possible.

Diversification approaches in the literature can be described as implicit or explicit, in how they aim to understand the different possible aspects of a query (Santos et al., 2015). The implicit approaches solely exploit the candidate set, i.e., the features of the initially retrieved documents, for diversification. Instead, the explicit approaches assume that the query aspects have been inferred beforehand (say, using topical directories (Agrawal et al., 2009) or query reformulations (Santos et al., 2010a)) and aim to prioritize the coverage of these aspects while generating the final ranking. Earlier studies have consistently shown that when the aspects are available, the explicit methods outperform the implicit ones. More recently, explicit diversification methods have also emerged as a promising approach to reduce bias and enhance fairness in various search scenarios (e.g. (McDonald et al., 2019)), where it is reasonable to assume the availability of such query aspects (such as gender, ethnicity and age in a job search scenario (Zehlike et al., 2017)). Thus, given the success of explicit diversification methods in typical search scenarios and their usage in new application areas, we argue that exploring ways of further improving their effectiveness is an important and timely research direction.

1.1. Research Questions

The key research question tackled in this paper can be formulated as "How can we exploit supervised learning methods to improve the effectiveness of explicit search result diversification?". To this end, we identify three sub-questions as follows:

- RQ1: How can we employ supervised learning, namely, typical learning to rank (LTR) algorithms, for explicit diversification?
- RQ2: Instead of learning a model for generating a diversified ranking, how can we learn a model to predict the importance of query aspects, to be used in a traditional (unsupervised) diversification method?
- RQ3: How can we cast the diversification problem into a fusion problem, and then adapt supervised learning methods to solve the latter?

To investigate answers to these questions, we build three different frameworks, each of which leverages supervised learning with different features and goals, as follows:

- *LTRDiv:* First, we devise features that capture the aggregated relevance of a candidate document to all query aspects to train a model via typical learning to rank (LTR) algorithms. This framework, LTRDiv, is intended to rank higher the documents relevant to multiple aspects and thus, it aims to maximize the diversity of the ranking (as captured with the intent-aware precision metrics (Agrawal et al., 2009), which will be discussed later).

- *AspectRanker:* In our second framework, AspectRanker, we apply supervised learning methods for a sub-task of the explicit diversification process, namely, predicting the importance of the aspects for a user query. While most diversification methods in the literature employ aspect importance as a key component, they usually assume a uniform distribution (i.e., all aspects are equally important) or a popularity-based instantiation obtained from external resources (which may not match the actual representation of aspects in the candidate document set). In our approach, we first re-rank the *candidate documents* for each aspect and employ several pre- and post-retrieval query performance predictors, *QPP*s (e.g., (Carmel & Yom-Tov, 2010)), to estimate how well a given aspect is covered in the candidate set. Then, we train models to rank the aspects for a given query. Finally, we map the aspect rankings to importance values and exploit them in an traditional (unsupervised) diversification method, namely, xQuAD (Santos et al., 2010a).

- *LmDiv:* Inspired by the LambdaMerge (*Lm*) method that aims to combine rankings for different query suggestions (Sheldon et al., 2011), we train models that merge rankings (of candidate documents) for each query aspect. In this case, the trained model (a neural network as in (Sheldon et al., 2011)) captures both the relationships of documents and aspects, as well as each aspect's importance, which is again estimated based on QPPs.

1.2. Novel Contributions

Our novel contributions in the context of each framework can be summarized as follows:

- Via our LTRDiv framework described above, we introduce a simple yet effective approach to apply typical LTR algorithms in the diversification context. To the best of our knowledge, our present paper is the first work that casts the diversification problem as that of learning a ranking model, which is based on the aggregated relevance of each document to all aspects of a given query. Our proposed approach is motivated by the observation that, theoretically, a ranking where documents are sorted with respect to the number of aspects they are relevant to (i.e., *covered* aspects) will maximize the well-known intent-aware diversity metrics (under certain assumptions that will be discussed later), and such a ranking is also likely to satisfy the users' diversity requirements in practice[1]. Therefore, we train models to predict

[1] Our LTRDiv (and LmDiv) frameworks focus on the *coverage* of the aspects but disregard the *novelty*, i.e., they neglect the redundancy among the covered aspects. This choice is justified by an earlier work (Santos et al., 2012) that has shown that coverage-based approaches outperform the novelty-based ones for diversification. Alternatively, our AspectRanker framework utilizes xQuAD, a hybrid approach employing components for both coverage and novelty.

the number of covered aspects *per document*, in isolation, while the closest works in the literature (e.g., (Zheng et al., 2017)) assume a sequential selection model (i.e., considering the previous documents selected into the ranking).

- For the first time in the literature, we propose to *learn* the importance of aspects by re-ranking the candidate set for each aspect and leveraging QPPs. In the closest work to ours in the literature, Ozdemiray & Altingovde (2014) directly employed such QPP estimates as the aspect importance values, but they did not explore the idea of training a model to predict these values, as we do in the AspectRanker framework.

- We adapt the LambdaMerge approach to search result diversification casting the diversification problem as a fusion task, namely the supervised merging of rankings per query aspect. Again, earlier works only considered traditional (unsupervised) merging strategies in this context (e.g., (Ozdemiray & Altingovde, 2015)), but did not exploit supervised learning. In our adaptation of LambdaMerge, we learn the aspects' importance based on the representation quality of each aspect in the *candidate set* (as in AspectRanker), and we optimize an objective function that takes into account the relevance (of a document) to multiple aspects of a query.

1.3. Summary of Key Findings

We conduct thorough experiments using both `BM25` (i.e., initial retrieval results based on the well-known BM25 matching function) and the best-performing `TREC systems (a.k.a. runs)` (i.e., employing more sophisticated retrieval methods) for four topic sets employed in the TREC Diversity Task between 2009 and 2012. In addition to the non-diversified initial ranking, we employ two strong baselines, namely, xQuAD (which has been the best performing diversification method in several TREC campaigns (Santos et al., 2015) and xQuAD_{SR}, a variant of xQuAD that employs aspect importance values set as in (Ozdemiray & Altingovde, 2014).

Our findings over the `BM25 runs` reveal that the proposed frameworks, especially AspectRanker and LmDiv, outperform all three baselines in terms of various well-known metrics. We further evaluate LmDiv, our most effective framework, over the best-performing `TREC runs`, and again show promising strong gains over the baselines (i.e., up to 18.9% w.r.t. xQuAD and 11.6% w.r.t xQuAD_{SR} for α-nDCG@2). These results suggest that learning a model for obtaining aspect importance values and scoring documents, simultaneously, is the most effective approach for applying supervised learning in explicit search result diversification. Overall, the success of our proposed frameworks in this paper reveal that explicit diversification performance can be considerably improved using supervised learning approaches that do not require very large training sets and/or excessive computing resources (contrary to the popular deep learning paradigm of our day), and hence, they are applicable in real life scenarios that require diversity (or fairness) among the results of a search system.

The remainder of this paper is structured as follows: In the next section, we provide preliminaries required in some approaches in areas relevant to our work, such as explicit diversification, query performance prediction and supervised learning for ranking or merging the results. Section 3 presents our three frameworks for diversification, namely, LTRDiv, AspectRanker and LmDiv. The experimental setup and results follow

in Sections 4 and 5, respectively. After presenting our methods and findings, in Section 6, we provide a more general overview of the related work to highlight the difference of our approaches from the others. Finally, we provide concluding remarks and future research directions in Section 7.

2. Background and Preliminaries

Our work builds on and/or incorporates various previous approaches, such as explicit diversification (and especially, the xQuAD method), query performance predictors, and supervised algorithms for ranking and merging. In this section, we briefly review the methods that we employ and/or adopt in this work together with the corresponding notations. Later in Section 6, we provide a more general literature review, and highlight how our work differs from the previous works.

2.1. Explicit Result Diversification and xQuAD

Consider a query q with a set of known aspects $A_q = \{a_1, \ldots, a_m\}$ and a candidate set D including N documents that is initially retrieved for the main query q. The goal of diversification is to obtain a final ranking R (with $|R| = k$, where k is usually less than N) that is both relevant to the query and diverse (i.e., covering as many and diverse aspects as possible).

One of the most successful explicit diversification approaches is xQuAD (Santos et al., 2010a), which was the top-performer in the Diversity Task of the TREC Web Track between 2009 and 2012 (e.g., see (Santos et al., 2015)). This is a greedy best-first approach that selects the document $d \in D$ that maximizes Eq. (1) in each iteration, until k documents are inserted into R.

$$Score(q, d, R) = (1 - \lambda) \Pr(d|q) + \lambda \sum_{a \in A_q} \Big[\Pr(a|q) \Pr(d|a) \prod_{d_j \in R} (1 - \Pr(d_j|a)) \Big], \qquad (1)$$

In Eq. (1), $\Pr(d|q)$ and $\Pr(d|a)$ denote the score of a document with respect to the main query, or an aspect, and can be calculated using any effective document ranking approach, such as BM25 (Santos et al., 2010a). The first summand of Eq. (1) aims to capture the relevance of a candidate document d to the main query q, while the right hand side represents the diversity, based on the sum of the coverage of each aspect a ($\in A_q$) by d. In the latter computation, $\Pr(a|q)$ represents the importance of that aspect for the query, and, by default, is uniform across all aspects (Santos et al., 2010a). Furthermore, xQuAD discounts the score contribution of a document for a particular aspect (i.e., $\Pr(a|q) \Pr(d|a)$) by the probability that the aspect has been already well-covered by the documents selected *earlier* into R, represented by the product term $\prod_{d_j \in R}(1 - \Pr(d_j|a))$. This discounting mechanism aims to prioritize documents with high scores for the aspects that are not yet covered in R, and hence, enhances the novelty of the final ranking. The trade-off parameter λ is used to balance the relevance and diversity in the final ranking R.

In this paper, we employ xQuAD as a representative explicit diversification method in our AspectRanker framework (Section 3.2) and as a baseline method in our empirical comparative experiments (Section 5).

2.2. Query Performance Prediction

For a search system, it is important to be able to estimate the quality of the ranking obtained for a query, as it opens the way for several optimizations (such as applying more sophisticated retrieval methods, or suggesting alternative query formulations to the user). Therefore, various query performance predictors (QPPs) have been proposed in the literature (e.g., see (Carmel & Yom-Tov, 2010) for an overview). QPPs are broadly categorized as pre-retrieval or post-retrieval with respect to when the estimation can be done, i.e. before or after retrieval. Earlier works reported that both types of predictors are useful and the overall prediction performance may further improve by using different types of QPPs in combination (Carmel & Yom-Tov, 2010).

In this work, we exploit both pre- and post-retrieval QPPs as features to learn a model for predicting the aspect importance values (e.g., $\Pr(a|q)$ in Eq. (1)) in the AspectRanker and LmDiv frameworks[2]. In the following, we briefly review the QPPs employed to this end. In addition to the notations introduced in the previous section, we follow the notations used in (Ozdemiray & Altingovde, 2014), where C and $s_q(d)$ denote the underlying document collection and the relevance score of a document d with respect to query q (i.e., $\Pr(d|q)$ or $\Pr(d|a)$ in Eq. (1)), respectively; D_q^n represents the top-n ranking of candidate documents D (based on the relevance scores $s_q(d)$) for a query. Note that we employ all the QPPs also for the top-n rankings $D_{a_i}^n$ (where $a_i \in A_q$), i.e., for the ranking of candidate documents w.r.t. each aspect of a query (as will be discussed in Section 3).

2.2.1. Pre-retrieval Predictors

In this paper, we use the following two pre-retrieval predictors:

- **maxSCQ:** This predictor is motivated by the intuition that if the collection contains documents that are similar to the query, then this query is more likely to have a higher performance (Zhao et al., 2008). The similarity score (SCQ) is computed as:

$$SCQ(q) = \sum_{t \in q} ((1 + \ln(freq_{C,t})) \ln(1 + \frac{M}{M_t})) \quad (2)$$

  where M is the number of documents in the collection, $freq_{C,t}$ is the frequency of term t in the collection C, and M_t is the number of documents with term t. Zhao et al. (2008) identified a variant of SCQ called maxSCQ that was the most effective; maxSCQ is the SCQ score for the query term t that maximizes Eq. (2).

- σ_1**:** Zhao et al. (2008) conjectured that as the standard deviation of the query terms' weights increases, the retrieval system would identify relevant documents

[2] QPPs have been exploited as features for other tasks in the context of result diversification, such as classifying the aspect intent (Santos et al., 2011) and predicting the trade-off parameter λ (Santos et al., 2010b), which are clearly different from the way they are used in our AspectRanker and LmDiv frameworks.

readily, since documents would discriminate easily. This predictor sums the deviations over the query terms and thus reflects the variability of the query as a whole.

$$\sigma_1(q) = \sum_{t \in q} \sqrt{\frac{1}{M_t} \sum_{d \in C_t} (w_{d,t} - \bar{w_t})^2} \tag{3}$$

$$\bar{w_t} = \frac{\sum_{d \in C_t} w_{d,t}}{|C_t|}, \tag{4}$$

where C_t is the set of documents including the query term t while $w_{d,t}$ denotes the weight of query term t in document d (calculated via tf-idf in Zhao et al. (2008)).

2.2.2. Post-retrieval Predictors

In this paper, we use various post-retrieval predictors:

- **Weighted Information Gain (WIG):** This QPP estimates retrieval effectiveness by computing the difference between the mean of relevance scores of documents in the ranking D_q^n, and the relevance score of the collection. The intuition behind this approach is that "high quality retrieval should be much more effective than just returning the average document" (Zhou & Croft, 2007). The WIG score is calculated as follows:

$$WIG(q) = \frac{1}{n\sqrt{l}}(avg_{d \in D_q^n}(s_q(d)) - s_q(C)), \tag{5}$$

  where $s_q(d)$ and $s_q(C)$ represent the relevance scores of documents $d \in D_q^n$ and the collection C, respectively. l is the length of the query q, and n is the size of the ranking.

- **Normalized Query Commitment (NQC):** Shtok et al. (2012) stated that using the mean of the relevance scores might be misleading, as the ranking may include non-relevant documents. Hence they propose a technique that measures the amount of deviation of the relevance scores of documents in the ranking w.r.t the mean score, and further normalized it w.r.t the collection score.

$$NQC(q) = \frac{\sqrt{\frac{1}{n}\sum_{d \in D_q^n}(s_q(d) - avg_{d \in D_q^n}(s_q(d)))^2}}{s_q(C)} \tag{6}$$

- **ScoreAvg:** ScoreAvg[3] has been used as a simpler form of WIG for fusion-based retrieval in (Markovits et al., 2012). Since they found that normalizing by the query length damages the quality of the prediction when using multiple lists, they employed sum normalization. That is, the relevance scores, $s_q(d)$, of documents in D_q^n are normalized so that they sum to 1.

- **ScoreDev:** ScoreDev[3] (Markovits et al., 2012), is similar to NQC, but instead of the normalization w.r.t the collection score, it applies sum normalization over the document scores.

Finally, the following three query performance predictors were introduced by Ozdemiray & Altingovde (2014).

- **ScoreRatio:** This predictor uses the idea that a higher score gap between the first and last documents in a ranking may imply a higher probability of non-relevant documents appearing in this ranking. It is calculated as follows:

$$ScoreRatio(q) = \frac{s_q(d_n)}{s_q(d_1)} \tag{7}$$

[3]The QPP was named this way by Ozdemiray & Altingovde (2014).

- **VScoreAvg:** For this QPP, it is assumed that there exists a virtual document that perfectly matches the query, as in (Ozdemiray & Altingovde, 2015). This virtual document d^V would contain only the terms in the query and it would have the average document length in the collection. Then, its score $s_q(d^V)$ is used to normalize the scores $s_q(d)$ (of $d \in D_q^n$) to obtain a predictor based on the mean relevance of the ranked documents, as follows:

$$VScoreAvg(q) = \frac{1}{n} \sum_{d \in D_q^n} s_q^{virtual}(d) \tag{8}$$

$$s_q^{virtual}(d) = \frac{s_q(d)}{s_q(d^V)} \tag{9}$$

- **VScoreFirst:** The score of the first document in the ranking (after normalization by the score of the aforementioned virtual document) is used as a query performance predictor.

2.3. Supervised Learning for Ranking

Modern search engines apply a two stage ranking where first an initial retrieval result is obtained using relatively efficient methods (such as BM25) and then a complex and expensive machine learnt ranker is applied over the latter set. To train such specialized models for the ranking task, several algorithms have been introduced in the last two decades (see (Liu, 2009) for an overview).

In a nutshell, the input for a LTR algorithm is an instance vector that is created for each document retrieved for a query, and includes features that capture the query-document matching (e.g., tf-idf, BM25, etc. scores) as well as document-quality features (PageRank, in/out degrees, etc.) and query features (e.g., see (Liu, 2009; Macdonald et al., 2012)). During training, the target label is the graded relevance judgment of a document for the query (usually obtained from the human assessors). The LTR algorithms can consider these labels on their own, in pairs or as a list, giving way to pointwise, pairwise and listwise learning approaches.

In this work, we leverage LTR approaches in two ways: First, in the LTRDiv framework, we cast the diversification problem to a LTR problem with appropriate features and a target feature that is intended to optimize the aspect coverage. Secondly, in the AspectRanker framework, instead of ranking documents, we rank the aspects of a query to infer their importance values. In both cases, we employ two representative LTR algorithms, namely, SVMRank and Random Forests.

2.4. Supervised Learning for Result Merging

A well-explored topic in the literature is merging query results that are obtained by using different retrieval methods (e.g. tf-idf, BM25, LTR, etc.) over the same corpora and/or over different collections. In this work, we adopt a particular supervised strategy, LambdaMerge, which has been proposed to merge the rankings that are obtained for a query and its reformulations (Sheldon et al., 2011). Next, we review LambdaMerge and a follow-up variant, and discuss our application of this method to the result diversification problem in Section 3.

Assume that a list of top-N results is retrieved for a query and each of its reformulations. The LambdaMerge method consists of two neural networks: the first one, called scoring network, employs features, x_d^r, capturing the relationship between a query reformulation and the document; while the second network, namely a gating network, uses features, z^r, that represent the quality of each reformulation. The total score of a document is generated by the co-operation of these two networks. The contribution of each reformulation measured by the gating network is multiplied by the score of the document passing through the scoring network, and adding all of them yield the final score of a document. This process is formulated as follows:

$$score_d = \sum_r \psi_r f(x_d^r, \gamma) \tag{10}$$

$$\psi_r = \text{softmax}(z^i, \delta) \qquad for\ i = 1, ..., r. \tag{11}$$

where r is a query reformulation, ψ represents the quality of the reformulation, which is computed by Eq. (11), x_d^r is the feature vector of document d over the result list of reformulation r, γ and δ are the weight matrices. f is the key function of the scoring network. While this function can be any differentiable function, Sheldon et al. (2011) chose a fully connected two-layer neural network.

Training a neural network that optimizes an evaluation metric (such as Normalized Discounted Cumulative Gain (nDCG)) is a challenge, due to the discontinuity of the metric. Therefore, LambdaMerge has adopted the approach of LambdaRank (Burges et al., 2006), which overcomes this problem by using a smoothed version of the objective. In particular, the gradients of the objective, namely O, for the parameters of score and gating networks are computed by applying the chain rule. The most crucial component is the partial derivative of O w.r.t. the score (generated by the output layer of network), $\frac{\partial O}{\partial score_d}$, which can be computed for various information retrieval (IR) evaluation metrics as follows:

$$\frac{\partial O}{\partial score_d} = \sum_i \mid \Delta_{di} \mid (\Upsilon_{d>i} - 1/(1 + e^{score_i - score_d})) \tag{12}$$

where i indicates every document in the ranking, and $\mid \Delta_{di} \mid$ is the difference value in the metric when documents i and d are swapped in the ranking; $\Upsilon_{d>i}$ is an indicator that shows which document is more relevant according to the relevance judgements. It is 1 if the document d is more relevant than i, and 0 otherwise. In LambdaMerge, the authors employ the nDCG metric while computing $\mid \Delta_{di} \mid$ (see (Sheldon et al., 2011) for further details). For the other derivatives, the traditional back-propagation scheme is applied, as for the original LambdaRank approach (Burges et al., 2006).

Finally note that Lee et al. (2015) extended the LambdaMerge architecture by increasing the number of hidden layers in the scoring neural network (as well as injecting additional feature types). Following the naming in the latter work, we refer to our diversification framework that is based on the original LambdaMerge as LmDiv-Shallow, while we denote the multi-layer version as LmDiv-Deep.

3. Supervised Learning for Explicit Search Result Diversification

In this paper, we exploit various supervised learning methods (i.e., LTR algorithms and neural networks) to either generate a diversified final ranking (in the LTRDiv and LmDiv frameworks), or to obtain aspect importance values to be used in combination with a traditional explicit diversification method (in the AspectRanker framework). In the following, we discuss the details for each of these frameworks.

3.1. The LTRDiv Framework

In this section, to answer our first research question, RQ1 in Section 1.1, we cast the result diversification problem into a typical ranking problem to exploit the existing LTR algorithms to obtain the final ranking. As discussed in Section 2.3, in the traditional setup for LTR, the goal is to maximize the relevance of the ranking for a given query; therefore, during training, the model learns to predict the relevance label of each document for a given query. In this case, the document is represented with the features that capture the query-document matching (e.g., tf-idf, BM25, etc. scores) as well as the document-quality (PageRank, in- and out degree, etc.).

To be able to use a LTR algorithm for diversification, we extend this setup. In particular, we describe both the features and the learning target to capture the relevance of a document not only to the main query, but also to its multiple aspects. To formally define the features and target label, assume a query q (i.e., the main query issued by a user) with a set of known aspects $A_q = \{a_1, \ldots, a_m\}$ and a candidate set D (i.e., the top-N documents retrieved for q). We re-rank the candidate set D for each aspect a_i, using a typical retrieval mechanism (e.g., tf-idf, BM25, etc., as will be discussed later). We denote each such ranking as $D^n_{a_i}$, where $n \leq N$. Intuitively, a document that is ranked at a higher position (i.e., close to the top) in many of these rankings $D^n_{a_i}$ is likely to be relevant to several aspects, and hence, would contribute positively to the diversity of the final ranking. Therefore, we represent a document d's coverage of the aspect set A_q with the features that compute the minimal, maximal and average rank (and score) of d over the rankings $D^n_{a_i}$ for $a_i \in A_q$, shown as follows:

$$f_d = \langle s(d,q), r(d,q), \max_{1\leq i\leq m} s(d,a_i), \frac{1}{m}\sum_{i=1}^{m} s(d,a_i), \min_{1\leq i\leq m} s(d,a_i), \\ \max_{1\leq i\leq m} r(d,a_i), \frac{1}{m}\sum_{i=1}^{m} r(d,a_i), \min_{1\leq i\leq m} r(d,a_i)\rangle \tag{13}$$

where $s(d,a_i)$ is the relevance score of document d for aspect a_i (sum-normalized over $D^n_{a_i}$), and $r(d,a_i)$ is the rank of document d in $D^n_{a_i}$. Note that the feature vector f_d also includes $s(d,q)$ and $r(d,q)$, i.e., the relevance score and rank of d for the main query q. In this manner, the feature vector for a document is capable of representing the relevance of a document to both the main query and its aspects. The target label for a document d is the number of covered aspects, i.e., those a document is stated to be relevant for (with a non-zero grade in the ground truth relevance judgments).

We summarise the LTRDiv framework in Fig. 1. For the learning component, any LTR algorithm from the literature can be employed, and we discuss the ones employed in this work in Section 4. Finally, we justify our choice of the learning target (the number of covered aspects by a document) in this framework by the following observation.

Table 1
Relevance of documents to query aspects for a toy scenario.

	d_1	d_2	d_3	d_4	d_5
a_1	1	1	0	0	1
a_2	0	1	0	0	1
a_3	0	1	0	1	0

Proposition 1 Assuming all query aspects are equally important and the relevance of each document to each aspect is binary, a ranking of N documents based on the number of covered aspects will yield the optimum scores for the precision oriented intent aware (IA) metrics such as Precision-IA (P-IA) and Discounted Cumulative Gain (DCG)-IA (Agrawal et al., 2009), for any cut-off value $k \leq N$.

Example. Let's consider a query q with the candidate set $D_q = \{d_1, d_2, d_3, d_4, d_5\}$. The query has 3 aspects, and the (binary) relevance ($rel_a(d)$) of each document to each aspect is given in Table 1. We want to retrieve the top-2 documents for q.

In this case, assuming all aspects are equally important (i.e., $Pr(a|q) = 1/m$ for a query with m aspects), a ranking R of the documents that is based on the covered aspects, namely, $d_2(3), d_5(2), d_1(1), d_4(1), d_3(0)$, maximizes both P-IA and DCG-IA for any rank cut-off value, and hence, the top-2 list includes $\{d_2, d_5\}$. For P-IA, shown in Eq. (14), this is easy to see, by taking the (constant) multiplication $1/m \times 1/k$ out of the summations and then swapping the order of summations. Then, for each rank position, covering the maximum number of aspects would give the optimal P-IA score, and the ranking $\{d_2, d_5\}$ (covering 3+2 aspects) is optimal. Applying the same transformations for the DCG-IA metric in Eq. (15), we again see the optimal ranking should provide the highest gains for each possible rank position and hence, for this example, the ranking $\{d_2, d_5\}$ is optimal. Note that, Chapelle et al. (2011) also noted that DCG-IA could be optimized by sorting the documents w.r.t. the expected gain, and under the aforementioned assumptions, this means a ranking with respect to the number of covered aspects, as given in Proposition 1. Since the above example conveys the intuition underlying Proposition 1, we omit a formal proof for reasons of brevity.

$$\text{P-IA@k} = \sum_{a=1}^{m} Pr(a|q) \frac{1}{k} \sum_{j=1}^{k} Rel_a(R_j) \tag{14}$$

$$\text{DCG-IA@k} = \sum_{a=1}^{m} Pr(a|q) \sum_{j=1}^{k} \frac{2^{Rel_a(R_j)}}{log(1+j)} \tag{15}$$

In light of the above discussion, we argue that training models to predict the number of aspects covered by each candidate document is a meaningful and promising learning target for our LTRDiv framework[4]. This also means that LTRDiv is a coverage-based approach and does not take novelty into account, i.e., it neglects the redundancy between

[4]We note that the model may underperform if the ground truth for the evaluation includes graded relevance judgments and/or non-uniform aspect importance values; yet in our experiments with graded relevance judgments, LTRDiv was still found to yield a good performance (see Section 5).

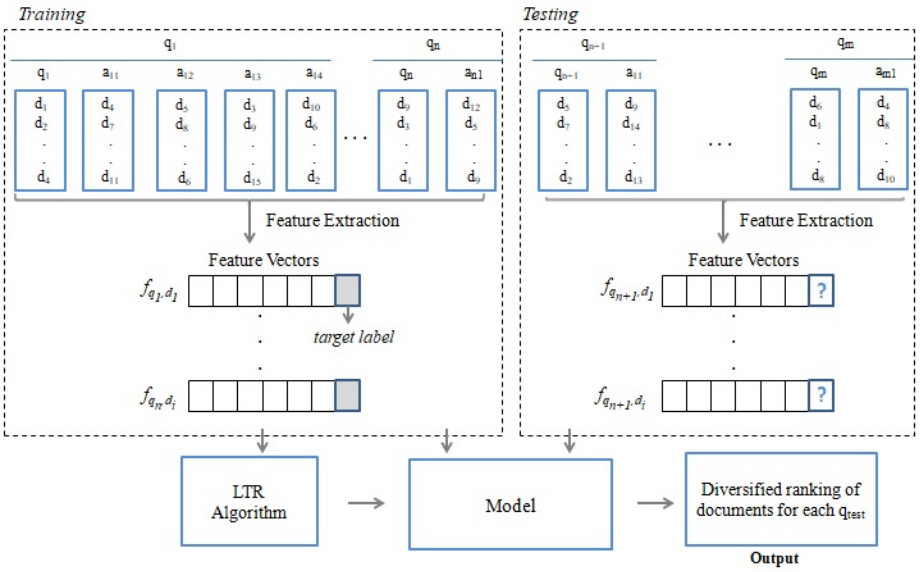

Fig. 1. LTRDiv framework to obtain a diversified ranking with a typical LTR algorithm.

the covered aspects. We note that this is a reasonable choice, since an earlier work has shown that solely targeting for the novelty is not effective for diversification (Santos et al., 2012), while coverage-based methods perform very well comparatively. In our experimental evaluation, we confirm the latter result and show that rankings obtained via LTRDiv yield high diversity scores, not only in terms of the intent-aware metrics, but also w.r.t. those taking novelty into account, such as α-nDCG.

3.2. The AspectRanker Framework

In this section, we propose the AspectRanker framework, as an answer to our second research question, RQ2 raised in Section 1.1. In the AspectRanker framework, our goal is to exploit the supervised learning methods, i.e., the LTR algorithms again, to estimate the aspect importance values that are employed in traditional (unsupervised) explicit diversification methods proposed in various earlier works (Agrawal et al., 2009; Santos et al., 2010a; Dang & Croft, 2012; Ozdemiray & Altingovde, 2015). In Algorithm 1, we specify the overall approach, in three stages, to obtain a diversified ranking. First, we train a model to rank the aspects for a given query. Next, we map each rank to a fixed importance value. Finally, we employ these importance values in a traditional diversification method. In the following, we describe in detail each stage.

For the ranking stage, we need to represent each query aspect with a feature vector and define a target label that would capture the importance of an aspect for a given query. To address this goal, we are inspired by an earlier work (Ozdemiray & Altingovde, 2014), which suggests that rather than relying on external resources for inferring aspect importance, one should consider to what extent an aspect is represented in the candidate set. For instance, for a given query "java", if the aspect "java island" is not covered adequately by any of the documents in the candidate set (i.e., all candidates are either relevant to the "programming language" or "coffee" aspects), then assigning

Algorithm 1: AspectRanker

Input : Q_{train} : the query set for training
Q_{test} : the query set for test
$\{D_1, .., D_Q\}$: candidate set retrieved for $q_i \in Q_{train} \cup Q_{test}$
$\{A_1, .., A_Q\}$: aspect set for $q_i \in Q_{train} \cup Q_{test}$
Output: R : set of diversified rankings R_i for q_i

for each training query $q \in Q_{train}$ **do**
  for each aspect $a_i \in A_q$ **do**
    Generate the ranking $D^n_{a_i}$ based on $s(d, a_i)$ for $d \in D_q$
    Construct feature vector $f_{a_i} = \langle \text{WIG}(a_i), \text{NQC}(a_i)... \rangle$
    Feed $\langle q, f_{a_i}, target \rangle$ triplets to train a LTR model
  end
end
for each test query $q \in Q_{test}$ **do**
  Obtain the aspect ranking A_q using the LTR model
  for each aspect $a_i \in A_q$ **do**
    Compute the importance of the a_i w.r.t its rank in A_q
  end
  Run xQuAD with computed aspect importance values
end

uniform importance values to all three aspects (following the common practice in the literature (Santos et al., 2010a)) would not help, but might even mislead the diversification process. To further justify our approach, in Fig. 2, we provide the percentage of aspects that have a given number of relevant documents in the top-100 candidate set, for the BM25 and TREC runs over 198 queries (described in detail in Sec. 4). Remarkably, a considerable percentage of aspects do not have even a single relevant document retrieved in the candidate set. Similarly, as the plot shows, the number of relevant documents does fluctuate: for a large fraction of aspects there are 1 to 10 relevant documents; but aspects with much larger numbers of relevant documents also exist. Therefore, as in (Ozdemiray & Altingovde, 2014), we employ the QPPs to capture the retrieval effectiveness of the top-ranked documents (in the candidate set) for each aspect. However, different from the work of Ozdemiray & Altingovde (2014), we do not directly employ the estimates of a single QPP as the aspect importance values, but instead learn a model to combine the estimates of several QPPs[5].

Formally, following the notations in the previous section, we again obtain the rankings $D^n_{a_i}$ of the candidate set D for each aspect a_i of a query q. For each such ranking, we compute the QPPs described in Section 2.2, namely, WIG, NQC, ScoreAvg, ScoreDev, ScoreRatio, VScoreAvg, VScoreFirst, maxSCQ, and σ_1. As the target label for each aspect, we calculate the well-known Precision metric over $D^n_{a_i}$, as follows:

[5]Note that the combination of QPPs has been explored independently in earlier works (Carmel & Yom-Tov, 2010), but we are not aware of any application in the context of result diversification for aspect ranking.

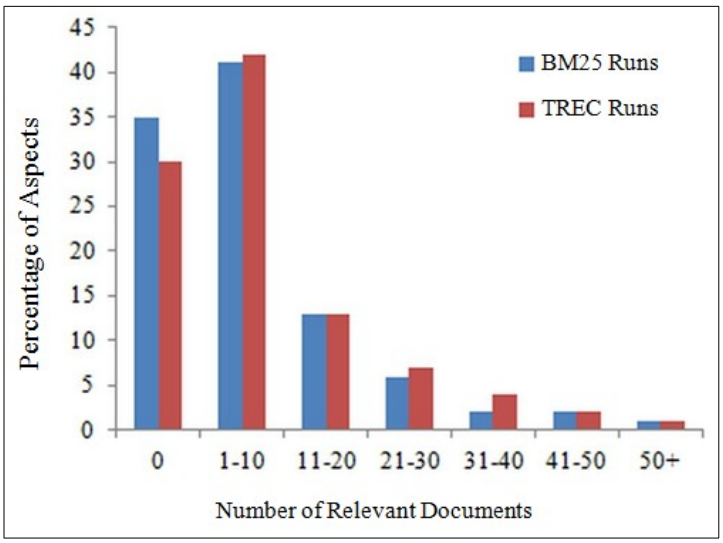

Fig. 2. Percentage of aspects (y-axis) with a given number of relevant documents (x-axis) in the candidate document sets (BM25 and TREC runs) for 198 queries.

$$Precision_{a_i,D} = \frac{|D^n_{a_i} \cap Rel_{a_i}|}{n} \tag{16}$$

where n ($\leq N$) is the size of the ranking for a_i and Rel_{a_i} is the set of documents judged relevant for the aspect a_i of the given query. Note that such a target label is an approximation, i.e., it is not guaranteed that using the target value as the aspect importance would maximize the diversification performance. However, the other alternative, trying all importance values (within a given range) for all aspects is prohibitively expensive, especially for queries with more than a few aspects. Hence, we opt to employ Eq. (16) as a proxy for the aspect importance to be predicted. These training instances are then fed to a LTR algorithm to learn a ranker for aspects.

During testing, once we obtain a ranking of aspects for a given query, we apply a simple procedure to map the ranks to actual importance values. We assign an aspect the importance value that is inversely proportional to its rank, i.e., for m aspects, the top-ranked one has the importance value m and the last one has 1. Note that one could also train a model, say using regression, to predict the $Precision_{a_i,D}$ value directly, but as discussed above, the target value in the model is a proxy for the actual aspect importance and hence, ranking aspects may produce more generalizable results than trying to predict an exact importance value. In our reported evaluation, we show that such an approximation yields indeed a very good performance. Finally, we use the estimated aspect importance values (after sum normalization) in an explicit diversification method to obtain a diversified ranking for a test query.

3.3. The LmDiv Framework

Our third research question, RQ3 raised in Section 1.1, is addressed in this section. We tailor the LambdaMerge approach (Sheldon et al., 2011) to search result diversifica-

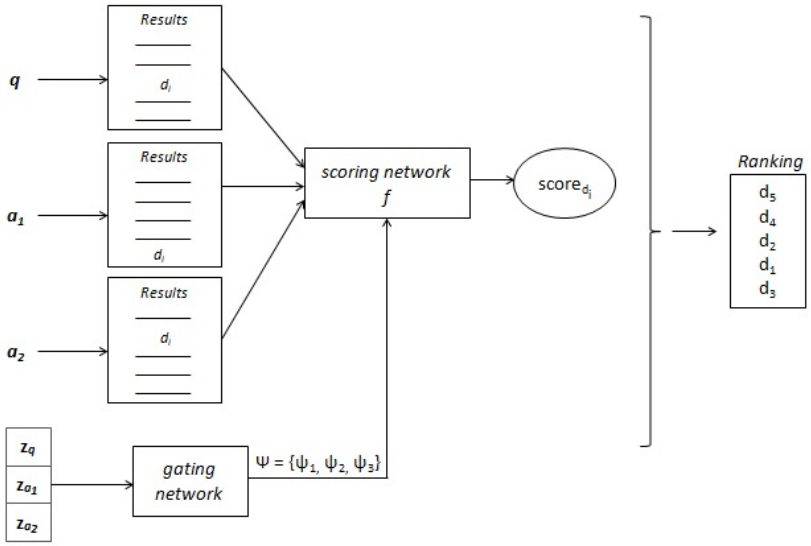

Fig. 3. The architecture of LmDiv (based on (Sheldon et al., 2011)).

tion casting the diversification problem as a fusion task, namely, the supervised merging of rankings per *query aspect*. Similar to LambdaMerge (reviewed in Section 2.4), our LmDiv framework simultaneously trains two neural networks, a scoring network that creates a score for each document-aspect pair, and a gating network that generates an importance value for each aspect (Fig. 3). The final score of a document is the weighted sum of the document-aspect scores with the corresponding aspect importance values, computed as follows:

$$score(d) = \sum_{a_i \in A} \psi_{a_i} h(f_{d,a_i}, \gamma) \tag{17}$$

$$\psi_{a_i} = \text{softmax}(z_{a_i}, \delta) \qquad for\ a_i \in A. \tag{18}$$

where a_i is a query aspect, ψ is the aspect importance, which is computed by Eq. (18), γ and δ are the weight matrices and $h()$ is the scoring neural network. In the following, we describe f_{d,a_i}, the feature vector of document d for aspect a_i, and z_{a_i}, the feature vector of aspect a_i, in detail.

Different from the LTRDiv framework, in this case, the feature vectors (f_{d,a_i}) are created for each $\langle d, a_i \rangle$ pair where $d \in D$, and $a_i \in A$ for a given query. Note that, in this case, we treat the original query q as an aspect and create all these features for q, as well. As before, $D^n_{a_i}$ denotes a ranking of candidate documents w.r.t. a_i. Then, following the practice in (Sheldon et al., 2011), we compute the following features for $\langle d, a_i \rangle$:

- $s(d, a_i)$: The raw relevance score of the document d for a_i
- $r(d, a_i)$: The rank of the document d in $D^n_{a_i}$
- SumNormScore: The relevance score $s(d, a_i)$ after applying sum normalization over all $d \in D^n_{a_i}$

- VirtualNormScore: The relevance score after applying virtual normalization (see Section 2.2 for the description of VScoreAvg (Ozdemiray & Altingovde, 2014)).
- StandardScore: The relevance score after applying z-score normalization.
- IsInTop: A binary indicator denoting whether the document is in $D^n_{a_i}$ (i.e., when $n < N$, some candidate documents would not appear among the top-ranked documents for a_i).

For the gating network, we again represent each aspect a_i using a feature vector, z_{a_i}, based on QPP scores over $D^n_{a_i}$, namely, WIG, NQC, ScoreAvg, ScoreDev, ScoreRatio, VScoreAvg, VScoreFirst, maxSCQ, and σ_1, as in the previous section. Hence, the network would learn the aspect importance values, which reflect the quality of each aspect's representation in the candidate result set.

During training, for each candidate document and aspect pair (d, a_i), the feature vectors f_{d,a_i} and z_{a_i}, are obtained and fed *simultaneously* to the scoring and gating networks, respectively, for each aspect a_i of the query; and their results are combined to obtain the final document score using Eq. (17). Once all documents for a query are scored, the LambdaRank gradients of the objective function is computed for the back-propagation.

The original LambdaMerge method aims to optimize the nDCG metric, as shown in Eq. (12), where the ground truth is simply based on the query-document relevance judgments. In this work, we define a different representation of the ground truth that is more appropriate for the diversification task. As in Section 3.1, for each document, we use the number of covered aspects as its target label. In this case, the $\Upsilon_{d>i}$ parameter in Eq. (12) returns 1 when a document d is relevant to a *larger* number of aspects than document i. Note that, such a formulation would learn a model to generate rankings in descending order of the number of covered aspects per document, as in the case of the LTRDiv framework, albeit using different features for the documents and aspects, and a neural network model optimizing a list-wise metric, i.e., nDCG. In this sense, LMDiv is also a coverage-based approach w.r.t. the categorization in (Santos et al., 2011).

As a further extension, we modify Eq. (12) to allow direct optimization of a metric specifically proposed for evaluating diversity, namely, nDCG-IA (Agrawal et al., 2009). In our adaptation, we compute the impact of swapping two documents for each aspect, separately, and then average over the aspects, as follows:

$$\frac{\partial O}{\partial score_d} = \sum_i \Pr(a|q) \sum_a \mid \Delta_{di} \mid (\Upsilon_{d>i} - 1/(1 + e^{score_i - score_d})) \tag{19}$$

In Eq. (19), $\Pr(a|q)$ denotes the aspect importance in the *ground truth* (if available), i indicates every document in the ranking, $\mid \Delta_{di} \mid$ is the difference value in the nDCG metric when documents i and d are swapped in the ranking, and $\Upsilon_{d>i}$ is an indicator that shows which document is more relevant (to an aspect a) according to the relevance judgements (which may be binary or graded). It is 1 if the document d is more relevant than i, and 0 otherwise.

Finally note that, any target metric used in this context must be *consistent* (Dinçer et al., 2014), i.e. an improving swap (where a document with a higher label moves above one with a lesser label) must result in a positive or 0 change in the metric. nDCG-IA

is also usable as the training objective for LmDiv, as it reduces to nDCG per aspect (captured in the inner summation of Eq. (19)) and nDCG is known to be consistent (e.g., (Dinçer et al., 2014)).

4. Experimental Setup

Dataset and runs. We employ query (topic) sets that have been developed for the the Diversity Task of the Text REtrieval Conference (TREC) Web Track between 2009 and 2012. Each set includes 50 queries (except for 2010, which has 48) along with their aspects and relevance judgments at the query and aspect levels. The candidate sets (i.e., the initial retrieval results per query, or shortly, *run*) are obtained over the ClueWeb09 Category B dataset, which consists of about 50 million English web pages (Clarke et al., 2009). For all diversification methods (those proposed and the baselines), we employ the official query aspects to isolate the evaluation of the diversification method from that of the aspect inference stage (as in several previous works, such as (Santos et al., 2010a; Dang & Croft, 2012; Ozdemiray & Altingovde, 2015)).

We present the performance of the proposed frameworks on two types of runs. `BM25 runs` are obtained by processing each query over the collection using our own retrieval system implementing the traditional BM25 model. We opt for BM25 because it is still the most widely used IR model in practical settings, and it does not require additional features (as in the LTR algorithms), enabling others to replicate our experiments. The parameters of BM25, namely k_1 and b, are set experimentally to 1.2 and 0.5, respectively. The documents with a spam percentile-score of 60 or lower according to the Waterloo Spam Rankings are eliminated from the results, following Cormack et al. (2011). We retrieve the top-100 documents for each query as the candidate set (i.e., N=100). Whenever needed, we apply sum normalization over the BM25 scores.

To be able to evaluate the diversification performance when the initial retrieval stage employs more sophisticated approaches beyond BM25, we also select the best runs submitted to previous TREC campaigns, which we refer to as `TREC runs`. While doing so, we only consider runs submitted to the ad hoc track (i.e., without any diversification method applied) over the ClueWeb09 Category B collection (following the practice in (Kharazmi et al., 2016; Akcay et al., 2017)). The best performing run is the one that yielded the highest α-nDCG@20 score for a given year. The ids of the selected runs for each year are as follows: Ucdsiftinter (2009), uogTrB67 (2010), Srchvrs11b (2011) and Qutparabline (2012). As in the previous case, we focus on the top-100 results from each run for diversification.

Note that, for the TREC runs, we do not have access to the retrieval methods employed to generate each of these runs. Therefore, for $s(d, q)$, i.e., the relevance score of a candidate document d for the main query q, we employ the score provided in the corresponding run (after normalization). To compute $s(d, a_i)$, the relevance of a document to an aspect a_i, we use BM25, following the practice in (Kharazmi et al., 2016; Akcay et al., 2017).

LTR algorithms. For the LTRDiv and AspectRanker frameworks, we experiment with two LTR algoritms, namely, SVMRank[6] and Random Forests (RF). The former is

[6] `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

a well-known pairwise LTR algorithm that optimizes the pairwise-loss over the training instances. We choose the second algorithm, RF, as a representative for the RankLib software package[7], as it has been shown to be the best LTR method among various competitors in a recent study (Deveaud et al., 2019). Our preliminary experiments also revealed that it is the best performing RankLib method in our setting.

Setup for the supervised learning frameworks. For all supervised methods, we apply 5-fold cross-validation to evaluate the performance. For LTRDiv, we train the LTR algorithms using the top-100 candidate documents for the training queries. In AspectRanker, we calculate the aspect features (i.e., QPPs) using the top-20 documents of the re-rankings, i.e., set $n = 20$ for D_q^n (since (Ozdemiray & Altingovde, 2014) suggested that considering only the top-ranked documents is adequate to determine the aspect representation quality in the candidate set). The diversification stage in AspectRanker employs xQuAD (Santos et al., 2010a), which is applied again over a candidate set of 100 documents. Finally, the LmDiv framework is trained with the re-rankings of the top-25 candidate documents per query aspect.

For the *shallow* version of the LmDiv framework, we train a fully connected two-layer neural network with four neurons in the hidden layer and a linear combination function as the output, as in (Sheldon et al., 2011). The multi-layer version (referred to as *deep* following Lee et al. (2015)) employs four hidden layers, each with fifteen neurons. Both neural networks are trained by stochastic gradient descent. In our preliminary experiments, the number of epochs and learning rate are determined as 25 and $5{\cdot}10^{-3}$, respectively, over the BM25 run for the TREC 2010 topic set, and the same values are employed for all the other runs and topic sets. We supply the training queries in a random order for each epoch. We batch the parameter updates by query for faster training as in (Burges et al., 2006). The neural networks optimize the nDCG and nDCG-IA metrics, as discussed in Section 3.3.

Baselines. In addition to reporting performance for the non-diversified (NonDiv) ranking, we apply two strong baselines. First, we use xQuAD with uniform aspect importance values[8] (as commonly employed in the literature (Santos et al., 2010a; Dang & Croft, 2012; Ozdemiray & Altingovde, 2015). Secondly, we employ an xQuAD variant where the aspect importance values are based on the estimations of a single QPP. In particular, we use the variant that employs the ScoreRatio predictor (see Section 2.2), as it is found to be the best-performing one in (Ozdemiray & Altingovde, 2014). We refer to the latter baseline as xQuAD_{SR}. For both of these baselines, the probabilities $P(d|q)$ and $P(d|a)$ are based on the respective sum-normalized relevance scores $s(d, q)$ and $s(d, a)$, of which computations are specified where the BM25 and TREC runs are described; and the trade-off parameter λ is determined using a 5-fold cross-validation.

Evaluation. We report results in terms of the widely used diversification metrics (see (Santos et al., 2015) for an overview), namely, α-nDCG, ERR-IA, Precision-IA, ST-recall, and MAP-IA, at the cut-off value of 20. Furthermore, we provide a more detailed picture for the α-nDCG metric at different cut-off values (2, 10 and 20), since this metric is capable of assessing both diversity and novelty in the results. Note that while certain

[7] `https://sourceforge.net/p/lemur/wiki/RankLib/`

[8] Note that, in our preliminary experiments, we also employed an alternative, popularity-based importance values (computed using the estimated number of results for each aspect from a major search engine) as in (Santos et al., 2010a), and verified that uniform values yield superior results.

Table 2
Diversification performance of the LTRDiv framework for the BM25 runs (over TREC 2009-2012 topic sets). The superscripts with (†) and (∗) denote a statistically significant difference from NonDiv and xQuAD at 0.05 level, respectively. For LTRDiv variants, % gains w.r.t. xQuAD_{SR} are shown in parentheses.

	BM25 Runs				
Method	ERR-IA@20	α-nDCG@20	P-IA@20	ST-Recall@20	MAP-IA@20
NonDiv	0.2626	0.3567	0.1558	0.5705	0.0351
xQuAD	0.3061	0.4089	0.1751	0.6199	0.0447
xQuAD_{SR}	**0.3309**	**0.4294**	0.1813	**0.6211**	0.0462
$\text{LTRDiv}_{\text{SVM}}$	$0.3188^{\dagger}$(-3.7%)	$0.4183^{\dagger}$(-2.6%)	$\mathbf{0.1908}^{\dagger,*}$(5.2%)	0.6146(-1.0%)	$\mathbf{0.0491}^{\dagger,*}$(6.3%)
$\text{LTRDiv}_{\text{RF}}$	$0.2984^{\dagger}$(-9.8%)	$0.4020^{\dagger}$(-6.4%)	$0.1812^{\dagger}$(-0.1%)	0.6168(-0.7%)	$0.0430^{\dagger}$(-6.9%)

methods try to learn and exploit aspect importance during the diversification stage, the evaluation stage assumes that all aspects are equally important, which is the common practice in the literature. All metrics are computed using the ndeval software[9]. We use the Student's two-tailed paired t-test (at 95% confidence level) for analyzing statistical significance.

5. Experimental Results

In this section, we first provide the evaluation results for all three frameworks using the BM25 runs. Next, for the best performing framework, LmDiv, we provide further results using the TREC runs.

5.1. Diversification Performance of Supervised Learning for the BM25 Runs

LTRDiv Framework. We begin with presenting the performance of our first framework, LTRDiv, presented in Section 3.1. Table 2 reports the performance of LTRDiv with two typical LTR algorithms, i.e., SVMRank and RF, in terms of diversity-aware metrics. As a first observation, we see that LTRDiv provides a notable improvement over the non-diversified BM25 baseline for all metrics. For instance, while NonDiv yields an α-nDCG score of 0.3567, LTRDiv with SVMRank yields 0.4183 and with RF it yields 0.4020.

For the majority of metrics, we observe that using SVMRank in LTRDiv is better than using RF. LTRDiv with SVMRank also outperforms the xQuAD baseline for all metrics except ST-recall, and performs better than the most-effective baseline, xQuAD_{SR}, for the P-IA and MAP-IA metrics. In the latter case, the improvements w.r.t. P-IA and MAP-IA metrics reach up to 5.2% and 6.3%, respectively, over xQuAD_{SR}. In short, we conclude that LTRDiv is better than NonDiv and a strong diversification baseline, xQuAD; but it can beat the strongest baseline, xQuAD_{SR}, for only two of the evaluation metrics. These results indicate that taking the aspect importance values into account is important for diversification (as the most effective approach in Table 2, xQuAD_{SR}, employs the ScoreRatio predictor for this purpose), and justify our motivation for the AspectRanker and LmDiv frameworks, both of which aim to predict such importance values via supervised learning.

[9] http://trec.nist.gov/data/web10.html

Table 3
Diversification performance of the AspectRanker framework for the BM25 runs (over TREC 2009-2012 topic sets). The superscripts with (†) and (∗) denote a statistically significant difference from NonDiv and xQuAD at 0.05 level, respectively. For the AspectRanker variants, % gains w.r.t. xQuAD_{SR} are shown in parentheses.

	BM25 Runs				
Method	ERR-IA@20	α-nDCG@20	P-IA@20	ST-Recall@20	MAP-IA@20
NonDiv	0.2626	0.3567	0.1558	0.5705	0.0351
xQuAD	0.3061	0.4089	0.1751	0.6199	0.0447
xQuAD_{SR}	**0.3309**	0.4294	0.1813	0.6211	0.0462
$\text{AspectRanker}_{\text{SVM}}$	$0.3308^{\dagger,*}$(0.0%)	$\mathbf{0.4328}^{\dagger,*}$(0.8%)	$0.1849^{\dagger,*}$(2.0%)	$\mathbf{0.6314}^{\dagger}$(1.7%)	$\mathbf{0.0482}^{\dagger,*}$(4.3%)
$\text{AspectRanker}_{\text{RF}}$	$0.3297^{\dagger,*}$(-0.4%)	$0.4320^{\dagger,*}$(0.6%)	$\mathbf{0.1852}^{\dagger,*}$(2.2%)	$0.6290^{\dagger}$(1.3%)	$0.0470^{\dagger}$(1.7)%

Table 4
Diversification performance of the LmDiv framework for the BM25 runs (over TREC 2009-2012 topic sets). The superscripts with (†) and (∗) denote a statistically significant difference from NonDiv and xQuAD at 0.05 level, respectively. For the LmDiv-S (Shallow) and LmDiv-D (Deep) variants, % gains w.r.t. xQuAD_{SR} are shown in parentheses.

	BM25 Runs				
Method	ERR-IA@20	α-nDCG@20	P-IA@20	ST-Recall@20	MAP-IA@20
NonDiv	0.2626	0.3567	0.1558	0.5705	0.0351
xQuAD	0.3061	0.4089	0.1751	0.6199	0.0447
xQuAD_{SR}	0.3309	0.4294	0.1813	0.6211	**0.0462**
LmDiv-S	$0.3344^{\dagger,*}$(1.1%)	$0.4358^{\dagger,*}$(1.5%)	$\mathbf{0.1845}^{\dagger,*}$(1.8%)	$0.6226^{\dagger}$(0.2%)	$\mathbf{0.0462}^{\dagger}$(0.0%)
LmDiv-D	$\mathbf{0.3454}^{\dagger,*}$(4.4%)	$\mathbf{0.4410}^{\dagger,*}$(2.7%)	$0.1748^{\dagger}$(-3.6%)	$\mathbf{0.6250}^{\dagger}$(0.6%)	$0.0442^{\dagger}$(-4.3%)

AspectRanker Framework. Table 3 presents the performance of our second framework, AspectRanker, where our goal is to predict the aspect importance values to be used in a traditional (unsupervised) diversification algorithm, namely, xQuAD. Again, we employ the SMVRank and RF algorithms with AspectRanker. Our findings show that AspectRanker, with any of these LTR algorithms, outperforms (significantly) both the NonDiv and xQuAD baselines for all metrics (yielding relative gains of up to 26% (0.2626 → 0.3308) and 8% (0.3061 → 0.3308), over NonDiv and xQuAD, respectively, for ERR-IA). We also observe that employing SVMRank in AspectRanker yields a better performance than employing RF, for most of the metrics. More crucially, AspectRanker (with SVMRank or RF) can also beat the strongest baseline, xQuAD_{SR}. Specifically, AspectRanker (with SVMRank) yields relative improvements of 0.8%, 2%, 1.7%, and 4.3% for α-nDCG, P-IA, ST-Recall and MAP-IA metrics, respectively, in comparison to xQuAD_{SR}. These findings indicate that the supervised learning of aspect importance values provides higher performance improvements to the xQuAD method, in comparison to estimating these importance values using a single estimator, as in xQuAD_{SR}.

LmDiv Framework. In our last framework, LmDiv, we evaluate the impact of exploiting supervised learning for both determining the aspect importance values and generating the final document scores for ranking, simultaneously. In Table 4, we report the diversification performance of deep and shallow neural network architectures (described in

the experimental setup), referred to as LmDiv-Shallow and LmDiv-Deep, respectively. Note that we experimented with optimizing both of the nDCG & nDCG-IA metrics (as discussed in Section 3.3) for both cases; and found out that the results with the former metric are slightly better. As a consequence, we only report the latter results for the sake of saving space.

Table 4 shows that LmDiv-Shallow outperforms the baselines for all metrics but MAP-IA (for which the same score is obtained as xQuAD$_{SR}$), albeit with relative improvements less than 2%. LmDiv-Deep is not as effective as the diversification baselines for P-IA and MAP-IA, but yields the highest scores for the ERR-IA, α-nDCG and ST-Recall metrics; providing relative improvements of 12.8%, 7.8%, 0.8% over xQuAD, and 4.4%, 2.7%, 0.6%, over xQuAD$_{\text{SR}}$, respectively. Note that the gains of the LmDiv-Deep method over xQuAD (and also over NonDiv) are statistically significant. In comparison to xQuAD$_{\text{SR}}$, the gains are not identified as significant using a paired t-test, yet they are still numerically impressive, i.e., up to 4.4%.

We also note that the LmDiv-Deep approach is the overall winner for the ERR-IA and α-nDCG metrics (cf. Tables 2, 3 and 4). These two metrics address both the diversity and novelty of a ranking (as they have the diminishing return property, i.e., submodularity) and hence, they are seen as a better fit for assessing diversification effectiveness (Chapelle et al., 2011). Therefore, in what follows, we choose to provide results for α-nDCG, as a representative for the family of submodular metrics – which are used extensively in earlier works as well as in the Diversity Task of the TREC campaigns – at additional cut-off values for further insights.

Table 5 demonstrates the diversification performance of the LmDiv-Shallow and LmDiv-Deep methods for the α-nDCG evaluation metric at cut-off values of 2, 10 and 20 (the last column is repeated from Table 4 to facilitate comparisons). We see that both versions of LmDiv (i.e. Shallow or Deep) in Table 5 outperform xQuAD and xQuAD$_{\text{SR}}$ at all rank cut-off values. In particular, LmDiv-Deep achieves the best performance and provides a relative improvement of 7.2%, 3.2%, and 2.7% for α-nDCG@2, 10, and 20, respectively, over the strongest baseline, xQuAD$_{\text{SR}}$.

Next, we provide a gain/loss analysis to identify under what circumstances the LmDiv approach is more useful, i.e., provides gains over the baselines xQuAD and xQuAD$_{SR}$. To this end, since LmDiv aims to learn aspect importance values based on the evidence in the candidate document set D, we partition the query set according to the coverage of aspects there, as in (Dang & Croft, 2012). Specifically, we compute the ST-Recall scores over the BM25 runs (for our 198 queries), and split them into two groups, i.e., the queries for which the candidate set covered more than 50% of their aspects, and those covering less than 50%.

In the top two rows of Table 6, for each of these ST-Recall ranges, we present the percentage of queries that LmDiv (Shallow or Deep version) helps and hurts (shown as "Impr. Q."and "Hurt Q.", respectively), in terms of the α-nDCG scores, with respect to our first baseline, xQuAD. For each group of queries, we also present the overall improvement of the α-nDCG score, again over xQuAD (i.e., to show the total effect of the improved and hurt queries on the performance).

Our results show that, generally, a larger percentage of queries are improved than being hurt by LmDiv. We further observe that the percentages of both improved and hurt queries increase as the ST-Recall goes up. However, the increase for the improved queries is much higher, i.e., using our best performing LmDiv-Deep approach, the percentage

Table 5
Diversification performance (at different rank cut-off values) of the LmDiv framework for the BM25 runs (over TREC 2009-2012 topic sets). The superscripts with (†) and (∗) denote a statistically significant difference from NonDiv and xQuAD at 0.05 level, respectively. For the LmDiv variants, % gains w.r.t. xQuAD_{SR} are shown in parentheses.

	BM25 Runs		
Method	α-nDCG@2	α-nDCG@10	α-nDCG@20
NonDiv	0.2563	0.3214	0.3566
xQuAD	0.3117	0.3776	0.4089
xQuAD_{SR}	0.3444	0.3997	0.4294
LmDiv-S	$0.3484^{\dagger,*}$(1.2%)	$0.4067^{\dagger,*}$(1.7%)	$0.4358^{\dagger,*}$(1.5%)
LmDiv-D	$\mathbf{0.3693}^{\dagger,*}$(7.2%)	$\mathbf{0.4125}^{\dagger,*}$(3.2%)	$\mathbf{0.4410}^{\dagger,*}$(2.7%)

Table 6
The percentage of queries improved and hurt (in terms of α-nDCG) by the LmDiv variants over the baselines, xQuAD and xQuAD_{SR}, when queries are grouped by ST-Recall of the initially retrieved documents (BM25 runs over TREC 2009-2012 topics). The Score Impr. column presents the relative α-nDCG score improvement wrt. the corresponding baseline.

ST-Recall ranges	[0-0.5)			[0.5, 1]		
No of queries	65			133		
	Impr. Q	Hurt Q	Score Impr.	Impr. Q	Hurt Q	Score Impr.
LmDiv-S vs. xQuAD	32.31%	27.69%	5.14%	62.41%	34.59%	6.81%
LmDiv-D vs. xQuAD	30.77%	30.77%	2.85%	63.16%	34.59%	8.64%
LmDiv-S vs. xQuAD_{SR}	32.31%	27.69%	2.85%	52.63%	45.86%	1.27%
LmDiv-D vs. xQuAD_{SR}	26.15%	33.85%	0.61%	54.89%	42.86%	3.01%

of improved queries against xQuAD goes from 30.77% to 63.16% (more than doubled), while the percentage of hurt queries shows a small increase (from 30.77% to 34.59%). We also observe that for the queries with higher ST-recall, the score gains are higher (e.g., again for LmDiv-Deep, the relative score improvements against xQuAD is 2.85% for low-recall queries and 8.64% for high-recall queries). These findings indicate that the LmDiv approach is more useful when the candidate result set covers a reasonable number of query aspects.

In Table 6, the bottom two rows present a similar analysis against xQuAD_{SR}. Since the latter is a stronger baseline, the percentage of improved queries is lower for both query groups (in comparison to the xQuAD case), but the trend is similar, i.e., for LmDiv-Deep, the percentage of improved queries is again more than doubled (from 26.15% to 54.89%) comparing the low- and high-recall ranges.

Another question we seek to answer in this section is the impact of QPPs in the trained models, since several QPPs are employed as features to represent the aspects for the gating network component of the LmDiv framework (Fig. 3). In Table 7, we present the QPP features' weights in the best-performing LmDiv model trained for our BM25 run (over TREC 2009-2012 topics). As we have applied a 5-fold CV during training, for each year's query set, a feature's weight is the average weight over those obtained from the LmDiv models built for five different training folds. We also provide the overall

Table 7
The weights of QPP features in the LmDiv framework for the BM25 runs (over TREC 2009-2012 topic sets).

QPP Feature	2009	2010	2011	2012	Average
maxSCQ	-0.046	0.040	-0.062	0.003	-0.016
sigma	0.037	0.315	-0.083	0.173	0.111
WIG	0.362	0.414	0.145	-0.213	0.177
NQC	-0.116	-0.064	-0.003	0.062	-0.030
ScoreAvg	-0.031	-0.062	-0.014	0.200	0.023
ScoreDev	0.290	-0.046	0.033	0.315	0.148
ScoreRatio	-0.045	0.104	-0.056	0.083	0.021
VScoreAvg	0.373	0.274	0.100	0.307	0.263
VScoreFirst	0.440	0.401	0.213	0.260	0.328

average weight of a feature (over these 4 query sets) in the last column of the table.

Table 7 reveals that all features are likely to contribute positively in most of the cases. Specifically, the last two post-retrieval features, VScoreAvg and VScoreFirst, are consistently weighted higher, implying that they are more useful for the LmDiv models. Having said that, we also observe that the other features are found to be useful, and even the features that have negative weights on the average (maxSCQ and NQC) have positive weights for certain query sets. For this reason (and due to the fact that our models indeed have a moderate number of features, in comparison to the LTR models, which include hundreds of features), we prefer to keep all of them in our models. Note that this choice does not incur a significant efficiency overhead for diversification during the run-time, as we discuss next.

While our results up to this point demonstrate the superiority of the LmDiv framework in terms of diversification effectiveness, as we aim to propose approaches that are applicable in practical scenarios, we also provide a comparison of the algorithmic complexity of LmDiv and the baseline xQuAD approach.

To begin with, as discussed in Sec. 2.1, xQuAD constructs the final top-k ranking iteratively, by selecting the document that maximizes Eq. (1), which is computed for each document in the candidate set D in each iteration. Hence, the complexity of xQuAD is $\mathcal{O}(Nk)$, where $N = |D|$.

For the LmDiv framework, during diversification, the first step is preparing the feature vectors over the candidate set, which requires computing the QPP features. At this point, we would like to emphasize that xQuAD (like many other unsupervised methods, such as IA-Select, PM2, etc.) computes each candidate document's relevance score for each aspect (i.e., $s(d_j, a_i)$), hence the overhead of computing the post-retrieval QPPs (in comparison to xQuAD) is very low, and essentially amounts to obtaining the top-n ranking $D^n_{a_i}$ of these documents for each aspect (note that, as $n \approx k$ in our setup, we use k for simplicity in this analysis). Since k is smaller than N (by an order of magnitude, in some cases), we can obtain the rankings $D^k_{a_i}$ efficiently, by extracting the top-k documents of an aspect in $\mathcal{O}(k \log N)$ time using a size-N max-heap. Thus, the complexity of generating aspect rankings (to compute QPP features) for $|A|$ aspects is $\mathcal{O}(|A|k \log N)$. During the final score computation, LmDiv takes the feature vectors generated for each document and aspect pair as the input, implying $\mathcal{O}(N|A|)$ time complexity. Thus, the

Table 8
Diversification performance of the LmDiv framework for the TREC Runs (averaged over the four best-performing runs corresponding to TREC submissions between 2009-2012). The superscripts with (†), (∗), (‡) denote a statistically significant difference from NonDiv, xQuAD, xQuAD_{SR} at 0.05 level, respectively.

	TREC Runs				
Method	ERR-IA@20	α-nDCG@20	P-IA@20	ST-Recall@20	MAP-IA@20
NonDiv	0.3380	0.4452	0.1868	0.6564	0.0416
xQuAD	0.3709	0.4829	0.2091	0.6886	0.0511
xQuAD_{SR}	0.3851	0.4936	**0.2137**	0.6879	**0.0528**
LmDiv-S	**0.3950**†,*(2.6%)	0.5018†,*(1.7%)	0.2126†(-0.5%)	0.6850†(-0.4%)	0.0522†(-1.1%)
LmDiv-D	0.3913†,*(1.6%)	**0.5041**†,*(2.1%)	0.2040†(-4.5%)	**0.7086**†,‡(3.0%)	0.0517†(-2.1%)

overall complexity of LmDiv is $\mathcal{O}(N|A|)+\mathcal{O}(|A|k\log N)$; which is comparable to and even better than that of xQuAD, for practical values of N (between 50 and 1000), $|A|$ (at most 10) and k (at most 20), as also employed in the literature. In our experiments, we also observed that the run-time processing efficiency of LmDiv is better than that of xQuAD.

To summarize, our findings in this section indicate that learning a model for obtaining aspect importance values and scoring documents, simultaneously, is the most effective approach (especially, in terms of the ERR-IA and α-nDCG metrics addressing both diversity and novelty) for applying supervised learning in explicit search result diversification. Furthermore, as reported in other scenarios (Lee et al., 2015), a deeper neural network (i.e., with a larger number of hidden layers) is likely to yield a better performance than a shallow one. Overall, we conclude that our best-performing framework, LmDiv, is both effective and efficient for diversification.

5.2. Diversification Performance of the LmDiv Framework for the TREC Runs

To demonstrate the robustness of the best-performing framework, namely, LmDiv, we conduct additional experiments. To this end, we use the submitted runs that exhibit the highest α-nDCG@20 score in the ad hoc retrieval track of TREC between 2009 and 2012. Note that since these runs are not diversified, they can safely serve as the candidate sets (generated with various ranking methods beyond BM25), following the practice in (Akcay et al., 2017; Kharazmi et al., 2016).

In Table 8, we present the effectiveness of the LmDiv framework against the baselines by averaging the metric scores over the four different TREC runs (listed in Sec. 4). We observe that both LmDiv versions outperform all the baselines for the ERR-IA and α-nDCG metrics, while their performance is inferior to the strongest baseline, xQuAD_{SR}, for P-IA and MAP-IA (but again, better than NonDiv and/or xQuAD). In this case, there is no clear winner between the two LmDiv versions, yet LmDiv-Deep yields the highest relative improvements for the α-nDCG and ST-Recall metrics.

Next, as in the previous section, we focus on the α-nDCG metric at different rank cut-offs, and present results for each of the selected TREC runs in Tables 9-12.

Table 9 presents the diversification performance of LmDiv for the best-performing TREC 2009 run (*Ucdsiftinter*). In this case, LmDiv-Deep yields the highest scores for

Table 9
Diversification performance of the LmDiv framework for the best-performing TREC 2009 run, *Ucdsiftinter*. The superscripts with (†) denote a statistically significant difference from NonDiv at 0.05 level. For the LmDiv variants, % gains w.r.t. xQuAD_{SR} are in parentheses.

	TREC 2009 Run (Ucdsiftinter)		
Method	α-nDCG@2	α-nDCG@10	α-nDCG@20
NonDiv	0.2061	0.2534	0.2802
xQuAD	0.2519	0.2810	0.3010
xQuAD_{SR}	0.2833	0.2888	0.3116
LmDiv-S	**0.2996**†(5.7%)	0.2983†(3.3%)	0.3209†(3.0%)
LmDiv-D	0.2816†(-0.6%)	**0.3086**†(6.9%)	**0.3265**†(4.8%)

Table 10
Diversification performance of the LmDiv framework for the best-performing TREC 2010 run, *uogTrB67*. The superscripts with (†) denote a statistically significant difference from NonDiv at 0.05 level. For the LmDiv variants, % gains w.r.t. xQuAD_{SR} are in parentheses.

	TREC 2010 Run (uogTrB67)		
Method	α-nDCG@2	α-nDCG@10	α-nDCG@20
NonDiv	0.3378	0.3717	0.4178
xQuAD	0.3400	0.4146	0.4584
xQuAD_{SR}	0.3682	0.4325	0.4734
LmDiv-S	**0.3979**(8.1%)	**0.4434**†(2.5%)	**0.4822**†(1.9%)
LmDiv-D	0.3714(0.9%)	0.4359†(0.8%)	0.4754†(0.4%)

the top-10 and top-20 results, and again provides gains that reach up to 37% (0.2061 → 0.2816), 12% (0.2519 → 0.2816), 6.9% (0.2888 → 0.3086) over NonDiv, xQuAD and xQuAD_{SR}, respectively.

In Table 10, we report the results for *uogTrB67* from TREC 2010. The findings in this case are slightly different in that LmDiv-Shallow outperforms its Deep version. In particular, LmDiv-Shallow provides a relative improvement of 8.1%, 2.5% and 1.9% over xQuAD_{SR}, for cut-off values 2, 10 and 20, respectively. Nevertheless, both LmDiv approaches still outperform all the baselines for almost all cut-off values.

In Tables 11 and 12, we present our findings for the TREC 2011 and 2012 runs (with ids *Srchvrs11b* and *Qutparabline*, respectively). LmDiv-Deep is again the best-performing approach for diversification of both of these runs. According to Table 11, LmDiv-Deep provides relative gains of 11.6%, 4.3%, 4.1% over xQuAD_{SR}, for cut-off values 2, 10 and 20, respectively. Table 12 also reveals improvements, reaching up to 1.42% over xQuAD_{SR}.

Our findings in this section are important. In earlier works (Akcay et al., 2017; Kharazmi et al., 2016), it has been shown that providing an impressive diversification performance is more challenging when the initial retrieval results (i.e., NonDiv) are produced via sophisticated methods. Contrary to this, here we demonstrate that the LmDiv framework considerably improves the diversification performance (especially for the ERR-IA and α-nDCG metrics) over strong baselines (improvements being statistically

Table 11
Diversification performance of the LmDiv framework for the best-performing TREC 2011 run, *Srchvrs11b*. The superscripts with (†) and (∗) denote a statistically significant difference from NonDiv and xQuAD at 0.05 level, respectively. For the LmDiv variants, % gains w.r.t. xQuAD_{SR} are presented in parentheses.

	TREC 2011 Run (Srchvrs11b)		
Method	α-nDCG@2	α-nDCG@10	α-nDCG@20
NonDiv	0.4356	0.5312	0.5546
xQuAD	0.4630	0.5510	0.5747
xQuAD_{SR}	0.4657	0.5606	0.5872
LmDiv-S	0.5101*(9.5%)	0.5715(1.9%)	0.6033*(2.7%)
LmDiv-D	**0.5198**(11.6%)	**0.5848**†(4.3%)	**0.6111**†,*(4.1%)

Table 12
Diversification performance of the LmDiv framework for the best-performing TREC 2012 run, *Qutparabline*. The superscripts with (†) denote a statistically significant difference from NonDiv at 0.05 level. For the LmDiv variants, % gains w.r.t. xQuAD_{SR} are in parentheses.

	TREC 2012 Run (Qutparabline)		
Method	α-nDCG@2	α-nDCG@10	α-nDCG@20
NonDiv	0.4013	0.4943	0.5269
xQuAD	0.5011	0.5710	0.5963
xQuAD_{SR}	0.4971	0.5680	0.6014
LmDiv-S	0.5021†(1.0%)	0.5744†(1.1%)	0.6001†(-0.2%)
LmDiv-D	**0.5024**†(1.1%)	**0.5761**†(1.4%)	**0.6022**†(0.1%)

significant for NonDiv and/or xQuAD) and using the best-performing ad hoc runs from different years and research groups. Therefore, the findings in this section justify our goal of employing supervised learning in explicit diversification, and further reveal that a particular framework, LmDiv, is the most effective and robust approach to achieve this goal.

6. Related Work

In this paper, we leverage supervised learning to improve the performance of explicit search result diversification. In what follows, we position our work with respect to the literature on explicit diversification (and refer readers to the literature (e.g., (Santos et al., 2015)) for a detailed review of the implicit diversification methods (e.g., (Carpineto et al., 2012; Yu et al., 2018; Meng et al., 2018)) and in particular, supervised learning for the latter (e.g., (Yue & Joachims, 2008; Liang et al., 2014; Zhu et al., 2014; Xu et al., 2017). The discovery of explicit aspects (as in (Kim & Lee, 2015)) is also a related research direction that is not discussed here, as our methods are evaluated using the ideal aspects to isolate the effects of the discovery process.

The earliest work that devised a technique to exploit known query aspects is IA-Select, proposed by (Agrawal et al., 2009), which is similar to its successor xQuAD (Santos et al.,

2010a) but it lacked the relevance component in Eq. (1). The xQuAD method (described in Section 2.1) was reported to be the best method across several TREC campaigns, which motivated various optimizations over the original formulation (e.g., (Ozdemiray & Altingovde, 2015)). A more recent approach, PM-2, employs a strategy based on the allocation of seats to political parties in elections (Dang & Croft, 2012). Ozdemiray & Altingovde (2015) employed the aggregation of rankings (obtained for each query aspect) using unsupervised techniques, such as the well-known CombSum. All of these techniques are unsupervised, as they do not involve any training stage to learn a scoring function for diversification. Instead, the frameworks proposed in our work either directly learn a model (as in LTRDiv and LmDiv) to produce a diversified ranking, or learn a model to predict the aspect importance, which are used in all these prior approaches.

The LambdaMerge method adopted here is introduced by Sheldon et al. (2011) to improve the relevance of query results, and it merges rankings that are obtained over the entire collection for a query and its reformulations. In a follow-up study (Lee et al., 2015), LambdaMerge is also exploited for the fusion of results obtained over different collections or via different retrieval methods. However, to the best of our knowledge, LambdaMerge has not been applied for merging the re-rankings of the candidate documents for different query aspects, as we propose in this paper, for the purpose of diversification. Note that the LmDiv framework is close to a particular prior work, (Ozdemiray & Altingovde, 2015), since both are based on the idea of ranking aggregation; but the latter work employs unsupervised merging methods, whereas the LmDiv framework aims to *learn* a function to merge rankings.

All the aforementioned methods in the literature (namely, IA-Select, xQuAD, PM-2, CombSum based, etc.), require the aspect importance during diversification; however, most of the earlier works assume either a uniform probability distribution (Santos et al., 2010a; Ozdemiray & Altingovde, 2015) or set the aspects' importance using their popularity (say, in a collection (Santos et al., 2010a)). In a recent study, (Ozdemiray & Altingovde, 2014) proposed setting the aspects' importance values based on the score of a single QPP. Our AspectRanker framework extends the latter one in several ways: we use several QPPs at the same time as features, and learn a model to produce a ranking of the aspects, which is then mapped to the actual importance values.

Some earlier works showed that the normalization of relevance scores is important for the effectiveness of the unsupervised explicit diversification methods, and proposed alternatives, such as the so-called Virtual (Ozdemiray & Altingovde, 2015) and R60 (Zhang et al., 2018) normalization techniques. While we essentially employ the typical sum-based normalization here, our supervised methods may also benefit from incorporating such alternatives, which is a direction not further explored in this paper.

We are aware of only two works that have attempted to use supervised methods for explicit diversification. In the first one, Zheng et al. (2017) proposed a supervised method, L-HSRD, for hierarchical search result diversification. In this method, features are based on the relevance between aspects in each level of hierarchy and the candidate documents, and the model is trained using a sequential selection model (i.e., considering the previous documents in the ranking), as in (Zhu et al., 2014). Instead, the ranking functions learnt in the LTRDiv and LmDiv frameworks do not apply such a sequential selection process; i.e., they consider each document on its own during training and testing.

In the second work, Jiang et al. (2018) proposed a framework deploying a recurrent neural network with an attention mechanism. In particular, while scoring a new

document, the attention mechanism emphasizes the aspects that are not covered by the previously selected documents (i.e., following the sequential selection model discussed for L-HSRD). This is again different from the supervised learning employed in our proposed frameworks. In particular, both AspectRanker and LmDiv model the importance of an aspect based on its retrieval quality (over the candidate set) captured via QPPs, while the latter work weighs aspect(s) to "attend" at each iteration based on the previously selected documents.

7. Conclusions

In this paper, we sought an answer to the following key question: How can we exploit supervised learning methods to improve the effectiveness of explicit search result diversification? We identified three directions to achieve this goal, leading to three frameworks leveraging supervised learning with different features and goals. In the LTRDiv framework, we formulated the diversification problem as that of learning a ranking model, which is based on the aggregated relevance of each document to all aspects of a given query, and used well-known LTR algorithms, such as SVMRank and Random Forests. In our second framework, AspectRanker, we addressed a critical sub-task for result diversification and trained models (using various query performance predictors as features) to predict the importance of each query aspect. Then, these predicted values are exploited by a traditional (unsupervised) explicit diversification method. Finally, in the LmDiv framework, we adapted the LambdaMerge approach (Sheldon et al., 2011) for the supervised merging of rankings per query aspect.

Our exhaustive experiments over the standard TREC diversification topic sets (between 2009-2012) and using initial retrieval results generated by our group and other TREC participants justified the necessity and success of using supervised learning in this context. We found that all three frameworks can outperform a well-known diversification method, xQuAD, used as the baseline. Furthermore, determining the aspects' importance turned out to be a key factor, since only the AspectRanker and LmDiv frameworks that do so could outperform the strongest diversification baseline, i.e., a variant of xQuAD that also sets the aspect importance in an ad hoc manner. Overall, we found that learning a model for obtaining the aspect importance values and scoring documents, simultaneously, as in the LmDiv framework, is the most effective approach (especially, in terms of the metrics addressing both diversity and novelty) for applying supervised learning for the diversification problem.

The success of the proposed frameworks in this paper revealed that explicit diversification performance can be considerably further improved using supervised learning approaches that do not require very large training sets and/or excessive computing resources (contrary to the popular deep learning paradigm of our day), and hence, they are applicable in real life scenarios that require diversity (and even fairness, as in (Gao & Shah, 2020; McDonald et al., 2019)) among the results of a search system. In our future work, we plan to investigate ways of employing alternative diversification metrics as the objective function in the LmDiv framework. We also aim to adapt and evaluate our proposed frameworks in related yet different scenarios, such as in recommender systems.

8. Acknowledgements

We are grateful to Daniel Sheldon for providing the source code of LambdaMerge. This work is partially funded by Royal Society (UK) Newton Int.'l Exchanges Scheme (no. NI140231) and The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant no. 117E861. S. Yigit-Sert is supported by TÜBİTAK-BİDEB 2211/A program. I. S. Altingovde is partially supported by Turkish Academy of Sciences Distinguished Young Scientist Award (TUBA-GEBIP 2016). Last but not the least, we thank the anonymous reviewers for their valuable comments and suggestions.

References

Agrawal, R., Gollapudi, S., Halverson, A., & Ieong, S. (2009). Diversifying search results. In *Proceedings of the ACM WSDM* (pp. 5–14).

Akcay, M., Altingovde, I. S., Macdonald, C., & Ounis, I. (2017). On the additivity and weak baselines for search result diversification research. In *Proceedings of the ACM ICTIR* (pp. 109–116).

Burges, C. J. C., Ragno, R., & Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. In *Proceedings of the NIPS* (pp. 193–200).

Carmel, D., & Yom-Tov, E. (2010). *Estimating the Query Difficulty for Information Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers.

Carpineto, C., D'Amico, M., & Romano, G. (2012). Evaluating subtopic retrieval methods: Clustering versus diversification of search results. *Information Processing & Management (IPM)*, *48*, 358–373.

Chapelle, O., Ji, S., Liao, C., Velipasaoglu, E., Lai, L., & Wu, S. (2011). Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval Journal*, *14*, 572–592.

Clarke, C. L., Craswell, N., & Soboroff, I. (2009). Overview of the trec 2009 web track. In *Proceedings of TREC Conference*.

Cormack, G. V., Smucker, M. D., & Clarke, C. L. A. (2011). Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval Journal*, *14*, 441–465.

Dang, V., & Croft, W. B. (2012). Diversity by proportionality: an election-based approach to search result diversification. In *Proceedings of the ACM SIGIR* (pp. 65–74).

Deveaud, R., Mothe, J., Ullah, M. Z., & Nie, J. (2019). Learning to adaptively rank document retrieval system configurations. *ACM Transactions on Information Systems*, *37*, 3:1–3:41.

Dinçer, B. T., Macdonald, C., & Ounis, I. (2014). Hypothesis testing for the risk-sensitive evaluation of retrieval systems. In *Proceedings of the ACM SIGIR* (pp. 23–32).

Gao, R., & Shah, C. (2020). Toward creating a fairer ranking in search engine results. *Information Processing & Management (IPM)*, *57*.

Jiang, Z., Dou, Z., Zhao, X., Nie, J.-Y., Yue, M., & Wen, J.-R. (2018). Supervised search result diversification via subtopic attention. *IEEE Transactions on Knowledge and Data Engineering*, *30*, 1971–1984.

Kharazmi, S., Scholer, F., Vallet, D., & Sanderson, M. (2016). Examining additivity and weak baselines. *ACM Transactions on Information Systems*, *34*, 23:1–23:18.

Kim, S., & Lee, J. (2015). Subtopic mining using simple patterns and hierarchical structure of subtopic candidates from web documents. *Information Processing & Management (IPM)*, *51*, 773–785.

Lee, C.-J., Ai, Q., Croft, W. B., & Sheldon, D. (2015). An optimization framework for merging multiple result lists. In *Proceedings of the ACM CIKM* (pp. 303–312).

Liang, S., Ren, Z., & de Rijke, M. (2014). Personalized search result diversification via structured learning. In *Proceedings of the ACM SIGKDD* (pp. 751–760).

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, *3*, 225–331.

Macdonald, C., Santos, R. L. T., & Ounis, I. (2012). On the usefulness of query features for learning to rank. In *Proceedings of the ACM CIKM* (pp. 2559–2562).

Markovits, G., Shtok, A., Kurland, O., & Carmel, D. (2012). Predicting query performance for fusion-based retrieval. In *Proceedings of the ACM CIKM* (pp. 813–822).

McDonald, G., Thonet, T., Ounis, I., Renders, J.-M., & Macdonald, C. (2019). University of Glasgow Terrier team and Naver Labs Europe at TREC 2019 Fair Ranking Track. In *Proceedings of TREC Conference*.

Meng, Z., Shen, H., Huang, H., Liu, W., Wang, J., & Sangaiah, A. K. (2018). Search result diversification on attributed networks via nonnegative matrix factorization. *Information Processing & Management (IPM)*, *54*, 1277–1291.

Ozdemiray, A. M., & Altingovde, I. S. (2014). Query performance prediction for aspect weighting in search result diversification. In *Proceedings of the ACM CIKM* (pp. 1871–1874).

Ozdemiray, A. M., & Altingovde, I. S. (2015). Explicit search result diversification using score and rank aggregation methods. *Journal of the Association for Information Science and Technology*, *66*, 1212–1228.

Santos, R., Macdonald, C., & Ounis, I. (2010a). Exploiting query reformulations for web search result diversification. In *Proceedings of the WWW* (pp. 881–890).

Santos, R. L. T., Macdonald, C., & Ounis, I. (2010b). Selectively diversifying web search results. In *Proceedings of the ACM CIKM* (pp. 1179–1188).

Santos, R. L. T., Macdonald, C., & Ounis, I. (2011). Intent-aware search result diversification. In *Proceedings of the ACM SIGIR* (pp. 595–604).

Santos, R. L. T., Macdonald, C., & Ounis, I. (2012). On the role of novelty for search result diversification. *Information Retrieval Journal*, *15*, 478–502.

Santos, R. L. T., Macdonald, C., & Ounis, I. (2015). Search Result Diversification. *Foundations and Trends in Information Retrieval*, *9*, 1–90.

Sheldon, D., Shokouhi, M., Szummer, M., & Craswell, N. (2011). Lambdamerge: Merging the results of query reformulations. In *Proceedings of the ACM WSDM* (pp. 795–804).

Shtok, A., Kurland, O., Carmel, D., Raiber, F., & Markovits, G. (2012). Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems*, *30*, 11:1–11:35.

Xu, J., Xia, L., Lan, Y., Guo, J., & Cheng, X. (2017). Directly optimize diversity evaluation measures: a new approach to search result diversification. *ACM Transactions on Intelligent Systems and Technology*, *8*, 41.

Yu, H., Jatowt, A., Blanco, R., Joho, H., Jose, J. M., Chen, L., & Yuan, F. (2018). Revisiting the cluster-based paradigm for implicit search result diversification. *Information Processing & Management (IPM)*, *54*, 507–528.

Yue, Y., & Joachims, T. (2008). Predicting diverse subsets using structural svms. In *Proceedings of the ICML* (pp. 1224–1231).

Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., & Baeza-Yates, R. (2017). Fa*ir: A fair top-k ranking algorithm. In *Proceedings of the ACM CIKM* (pp. 1569–1578).

Zhang, Z., Xu, C., & Wu, S. (2018). Evaluation of score standardization methods for web search in support of results diversification. In *Proceedings of the WISA* (pp. 182–190).

Zhao, Y., Scholer, F., & Tsegay, Y. (2008). Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proceedings of the ECIR* (pp. 52–64).

Zheng, H.-T., Wang, Z., & Xiao, X. (2017). A learning approach to hierarchical search result diversification. In *Proceedings of the APWeb-WAIM* (pp. 303–310).

Zhou, Y., & Croft, W. B. (2007). Query performance prediction in web search environments. In *Proceedings of the ACM SIGIR* (pp. 543–550).

Zhu, Y., Lan, Y., Guo, J., Cheng, X., & Niu, S. (2014). Learning for search result diversification. In *Proceedings of the ACM SIGIR* (pp. 293–302).