Precision-Oriented Query Facet Extraction

Weize Kong and James Allan
Center for Intelligent Information Retrieval
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
{wkong, allan}@cs.umass.edu

ABSTRACT

Faceted search has been used successfully for many vertical applications such as e-commerce and digital libraries. However, it remains challenging to extend faceted search to the open-domain web due to the large and heterogeneous nature of the web. Recent work proposed an alternative solution that extracts facets for queries from their web search results, but neglected the precision-oriented perspective of the task – users are likely to care more about precision of presented facets than recall.

We improve query facet extraction performance under a precision-oriented scenario from two perspectives. First, we propose an empirical utility maximization approach to learn a probabilistic model by maximizing the expected performance measure instead of likelihood as used in previous approaches. We show that the empirical utility maximization approach can significantly improve over the previous approach under the precision-oriented scenario. Second, instead of showing facets for all queries, we propose a selective method that predicts the extraction performance for each query and selectively shows facets for some of them. We show the selective method can significantly improve the average performance with fair coverage over the whole query set.

Keywords

Query facet extraction; empirical utility maximization; performance prediction; selective query faceting

1. INTRODUCTION

Faceted search has been used successfully for many vertical applications such as e-commerce and digital libraries. However, while it holds great potential for assisting multi-faceted queries and exploratory search, it remains challenging to be extended to the open-domain web. The challenges stems from the large and heterogeneous nature of the web, which makes it difficult to generate facets for the entire web and recommend them for queries [24].

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2983323.2983824

To address the challenges, recent work [7, 15, 16] proposed an alternative solution that extracts facets for queries from their web search results, called query facet extraction. For example, when users search with the query "*baggage allowance*", the system might extract query facets like airlines, {*Delta*, *JetBlue*, *AA*, ...}, travel classes, {*first*, *business*, *economy*}, and flight types, {*international*, *domestic*}. Changing from a global approach that generates facets in advance for an entire corpus [23, 5] to a query-based approach that extract facets from the top-ranked search results, query facet extraction appears to be a promising direction for solving the open-domain faceted search problem – it not only make the facet generation problem easier, but also addresses the facet recommendation problem at the same time.

However, previous query facet extraction work neglected the precision-oriented perspective of the task, which could be important in practical use. As in many precision-oriented information retrieval tasks [12], we believe users are likely to care more about "precision" than "recall" for query facet extraction. That is, users may care more about the correctness of presented facets (*e.g.*, are the terms in the airline facet indeed about airlines, and are the airline terms grouped together in a same facet) than the completeness of facets (*e.g.*, are all possible facets for that query presented, and are all possible airline terms included the results?). In other words, users may prefer a few high-precision facets than many noisy facet results. Previous work [15, 7] did not consider this precision-oriented factor when designing query facet extraction models or evaluating extraction results. Therefore, it is unclear if these models can adapt to such scenarios.

In this paper, we study query facet extraction under the precision-oriented scenario. We first re-examine models and evaluation proposed in the previous work [15] under such scenario. For modeling, we find the learning objective used in the proposed model are not ideal for the task especially under the precision-oriented scenario. The proposed model is trained by maximum likelihood estimation on labeled training data. However, likelihood can be loosely related to the performance measure under the precision-oriented scenario. For evaluation, the work proposed a measure to combine precision and recall for facet terms, and the term clustering quality. We find that the measure double-counts the recall factor, which encourages generating large sized facets. We fix the issue and adopt the measure for facet evaluation under precision-oriented scenario by varying combining weights between the three factors.

Based on the our analysis, we improve query facet extraction performance under precision-oriented scenarios from two

perspectives. First, since likelihood can be loosely related to the performance measure in the precision-oriented scenario, we propose to directly maximizing the performance measure instead of likelihood during training using a empirical utility maximization (EUM) approach. However, exact optimization on the performance measure is difficult due to the non-continuous and non-differentiable nature of information retrieval measures. We address this problem by approximating the performance measure using its expectation. We show that this empirical utility maximization approach significantly improves over previous approaches under precision-oriented scenarios, suggesting utility is a better learning objective than likelihood, and our expectation-based approximation is effective.

Second, we improve extraction performance by a selective method that shows facets for good performing queries and avoid poor performing ones. We find that extraction performance varies for different queries – some queries are naturally more difficult than others for extracting query facets. In the precision-oriented scenario, it may be more desirable to avoid showing facets for those poor performing queries and leave the users with a clean keyword-search interface. A key problem, however, is how to predict the extraction performance. To solve this problem, we propose a simple and effective score based on the expectation of the performance measure. We find the score has a strong correlation with the performance measure, and when used in the selective method, it can significantly improve the average performance with fair coverage over the whole query set.

2. RELATED WORK

Open-Domain Faceted Search. Faceted search has been used successfully for many vertical applications such as e-commerce and digital libraries. However, it remains challenging to extend faceted search to the open-domain web due to the large and heterogeneous nature of its content, which makes it difficult to generate facets for the entire web and recommend them for queries [24]. Most of previous attempts on automatic facet generation [5, 19, 23, 21, 14, 18] were based on existing facet metadata or taxonomies, which are expensive to obtain or difficult to be extended to the web scale.

Query Facet Extraction. Our previous work [15] proposed query facet extraction – a technique that extracts facets for queries from their search results in order to explicitly represent interesting aspects of the queries. It was later adopted directly to solve the open-domain faceted search problem, and was shown to be a promising direction [16]. Changing from a global approach that generates facets in advance for an entire corpus [23, 5] to a query-based approach, query facet extraction not only makes the facet generation problem easier, but also addresses the facet recommendation problem at the same time. However, this direct adoption neglected the precision-oriented perspective of the task when used for open-domain faceted search. That is users may care more about correctness of presented facets than completeness of facets in practice. In this work, we address the issue and improve query facet extraction under the precision-oriented scenario. Other than faceted search, query facet extraction has also be used for diversifying search results [10, 9].

Directly Optimizing Performance Measures. Lots of previous work has proposed to directly optimize performance measures in learning for various information retrieval tasks, including ranking [28, 27, 3, 22, 6] and classification [20, 13, 11]. While higher performance is expected by doing so, it is usually difficult due to the non-continuous and non-differentiable nature of information retrieval measures. From the perspective of the loss function optimization, existing solutions fall into three categories [28]. First, one can minimize upper bounds of the basic loss function defined on the performance measures [27, 13, 31]. Second, one can approximate the the performance measures with functions that are easy to handle [11, 3]. Our work belongs to this category; it approximates the performance measure using a continuous function based on its expectation. Third, one can use specially designed technologies for optimizing the non-smooth performance measures [22, 6].

More related to our problem, Jansche proposed to train a logistic regression model by directly optimizing F-measures [11]. The work approximated the integer quantities in F-measures based on their probabilities, and thus made the optimization target continuous and differentiable. Then it trained the logistic regression model by optimizing the approximated F-measures on the training data. The method is also referred as empirical utility maximization (or empirical risk minimization) [29], which maximizes the expected utility (or performance) by its average utility on the training data as an approximation. The model and measure we study in this paper (see Section 4 and 5) is similar to Jansche's work, and we use similar approximation strategy in order to perform direct optimization on the performance measure.

Performance Prediction and Selective Methods. Previous work on performance prediction in information retrieval primarily focused on the core ranking problem. Many predictors/scores has been introduced for predicting retrieval performance, such as clarity score [4], average IDF [25] and robustness score [33]. Learning methods, such as regression models, has also been used to combine different factors for predicting retrieval performance [17, 2, 30].

One application of retrieval performance prediction is to allow the systems to invoke alternative retrieval strategies for different queries according to their performance. For example, Yom-Tov et al. [30] and Amati et al. [1] both showed that retrieval performance prediction can be used to improve the effectiveness of a search engine, by performing selective automatic query expansion for "easy" queries only. Our selective method for query facet extraction is similar as these methods in spirit – we want to selectively apply query facet extraction for "easy" queries only. To the best of our knowledge, no existing work has studied performance prediction for query facet extraction.

3. QUERY FACET EXTRACTION

In this section, we briefly review query facet extraction to give necessary background information for this work. Please refer to our original paper [15] for details.

3.1 Task Description

A **query facet** is a set of coordinate terms (*e.g.*, {*AA*, *Delta*, *JetBlue*,...}) that explicitly represent one aspect (*e.g.*, *airlines*) of its query (*e.g.*, *baggage allowance*). The coordinate terms share a semantic relationship by being grouped

under a more general hypernym ("is a" relationship). (*e.g.*, *AA*, *Delt*, *JetBlue* are all *airlines*). Terms in query facet are generally called **facet terms**. When it is clear from context, we will simply use "facet" for "query facet", and "term" for "facet term" for convenience. **Query facet extraction** is to extract query facets for a given query from certain resources, and in our case the top search results for that query.

3.2 General framework

Query facet extraction on search results generally works as follows: (1) Given a query, retrieve the top search results. (2) Extract candidate facets from the search results based on pre-defined extraction patterns. (3) Refine the candidates to final query facets by re-clustering facets or facet terms in the candidate set. Step 2 and 3 are further described below.

Candidate Extraction. We use the same two types of extraction patterns, textual and HTML patterns, as in our previous work [15]. For example, from the sentence "*... airlines such as AA, Delta and JetBlue*", according to the textual pattern "*term*, {,*terms*}*, *(and|or)* {*other*} *term*", we will extract the candidate facets {*AA*, *Delta*, *JetBlue*}. After the extraction, candidate facets are further cleaned [15] (*e.g.*, normalization, removing stopwords).

Facet Refining. The candidate facets extracted are usually very noisy [32], and could be non-relevant to the given query, therefore they need to be refined. This is the core problem of query facet extraction. Existing models differ in how they refine candidate facets, and our approaches aim to improve this part.

To help understand the refining problem, we show four candidate facets for the query *baggage allowance* in Table 1. L_1 contains terms that are relevant to *baggage allowance*, but they are not coordinate terms: *delta*, *france* and *round-trip* are not members of the same class. L_2 is a valid query facet, but it is incomplete – another airline *aa* appears in L_3. L_3 is mixed with different facets, airlines and travel classes. L_4 is non-relevant to the query.

Table 1: Four candidate facets for the query "*baggage allowance*"

L_1: delta, france, round-trip
L_2: delta, jetblue, british airways
L_3: aa, first, business, economy
L_4: hard to remember, playing it by ear, ...

The facet refining problem can be essentially viewed as a *selective* clustering problem, in which noisy terms are discarded, and facet terms (*e.g.*, *aa*, *delta*, *jetblue*, *british airways*, *first*, *business* and *economy*) are selected, then clustered into query facets (*e.g.*, {*aa*, *delta*, *jetblue*, *british airways*} and {*first*, *business*,*economy*}). Our previous work [15] proposed a graphical model for this problem, and showed it outperformed other existing methods. In this work, we develop the EUM approach based on the previous model, which we will re-examine in the next section.

4. QUERY FACETING MODEL

In this section, we briefly describe and re-examine query facet extraction model proposed in our previous work [15]. For convenience, we will refer the model QF (stands for query faceting) in the rest of this paper.

4.1 Problem Formulation

Before diving into the QF method, we first define the facet refining problem more formally. We use $F = \{t_i\}$ to denote a query facet, consisted by a set of facet terms t_i. $\mathcal{F} = \{F_i\}$ denotes the set of query facets for the given query. $T_{\mathcal{F}} = \bigcup_i F_i$ denotes all the facet terms in $\mathcal{F}$. Candidate facets are just an imperfect version of query facets, and we substitute "F" with "L" to denote corresponding variables. $L = \{t_i\}$ denotes a candidate facet. $\mathcal{L} = \{L_i\}$ denotes all the candidate facets. $T_{\mathcal{L}} = \bigcup_i L_i$ denotes all terms in the candidates. Then the facet refining problem is simply to find $\mathcal{F}$ constrained with $T_{\mathcal{F}} \subseteq T_{\mathcal{L}}$, given $\mathcal{L}$ (and possibly other resources).

In QF, facet refining problem was treated as a label prediction problem. It aims to learn and predict jointly 1) whether a candidate term is a facet term and 2) whether a pair of terms are in the same query facet. We denote the two types of labels as follows. The term labels are denoted by $Y = \{y_i\}$, where $y_i = 1\{t_i \in T_{\mathcal{F}}\}$ is a label indicating whether a candidate term t_i is indeed a facet term. Here $1\{\cdot\}$ is an indicator function which takes on a value of 1 if its argument is true, and 0 otherwise. The pair labels are denoted by $Z = \{z_{i,j}\}$, where $z_{i,j} = 1\{\exists\, F \in \mathcal{F},\, t_i \in F \wedge t_j \in F\}$ is a label indicates whether term t_i and t_j are in a same query facet. Thus, the facet refining problem is now formulated as the problem of predicting label Y and Z.

4.2 The Graphical Model

In QF, a directed graphical model is proposed for the labeling problem, aiming to capture the dependencies between the term and pair labels. We show the graphical model in Figure 1.

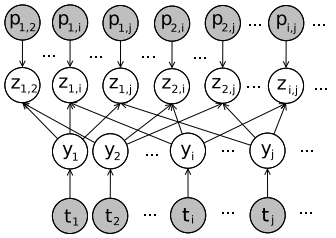


Figure 1: Query faceting graphical model

There are four types of variables. As defined before, Y and Z are the term and pair labels we aim to predict. $T_{\mathrm{L}} = \{t_i\}$ are all the candidate terms. $p_{i,j} = (t_i, t_j)$ is simply a short name for term pair t_i and t_j. $P_{\mathcal{L}} = \{p_{i,j} | \, t_i, t_j \in T_{\mathcal{L}},\, t_i \neq t_j\}$ denotes all the term pairs in $T_{\mathcal{L}}$. The conditional probability distributions are defined as:

$$P(y_i = 1 | t_i) = \frac{1}{1 + \exp\{-\sum_k \lambda_k f_k(t_i)\}}, \qquad (1)$$

$$P(z_{i,j} = 1 | p_{i,j}, y_i, y_j) = \frac{y_i y_j}{1 + \exp\{-\sum_k \mu_k g_k(p_{i,j})\}}, \qquad (2)$$

where f_k and g_k are features that characterize terms and term pairs respectively. We use the features proposed in the work [16] in our experiments. λ and μ are the weights associated with f_k and g_k, which need to be estimated during training. The joint conditional probability for the graphical model can be easily obtained by multiply all the conditional probability distributions. For more details, please refer to our previous paper [15].

4.2.1 Maximum Likelihood Estimation Training

A key problem of using QF for precision-oriented scenarios is that the model is trained by maximum likelihood estimation (MLE), which optimizes log-likelihood of given training dataset instead of the performance measure. The log-likelihood can be obtain as follows. Given a training set that contains ground truth labels for Y and Z, $\{T_{\mathcal{L}}^{(i)}, P_{\mathcal{L}}^{(i)}, Y^{(i)}, Z^{(i)}\}$, QF estimates model parameters λ and μ by maximizing the conditional log-likelihood of $\{Y^{(i)}, Z^{(i)}\}$ as

$$l(\lambda,\mu) = \sum_k \Big(\sum_i \log P(y_i^{(k)}|t_i^{(k)}) + \sum_{i,j} \log P(z_{i,j}^{(k)}|p_{i,j}^{(k)}, y_i^{(k)}, y_j^{(k)}) \Big), \tag{3}$$

where k indexes the training instances, i in the first term of the equation indexes all candidate terms, and i, j in the second term indexes all term pairs. With L2 regularization, the target can be reformulated as $l'(\lambda,\mu) = l(\lambda,\mu) - \frac{\sum_k \lambda_k^2}{2\sigma^2} - \frac{\sum_k \mu_k^2}{2\gamma^2}$. It can be shown that optimizing $l(\lambda,\mu)$ is equivalent as optimizing two independent logistic regression models, one for whether a candidate term is a facet term, and one for whether two *facet* terms should be clustered together [15].

While $l(\lambda,\mu)$ is straightforward to optimize, this training target can be loosely related to the extraction perform measure, especially in the precision-oriented scenario. Therefore, we use an empirical utility maximization approach to directly optimize the performance measure for training the same graphical model in Section 6.

4.2.2 Inference

We use the same inference algorithms as in the original QF method for predicting Y and Z labels and inducing facet output $\mathcal{F}$. We quickly review the algorithms as follows (see more details in the work [15]). Due to that maximum a posteriori inference on QF model is NP-hard, QFI and QFJ were proposed to approximate the results.

QFI approximates the results by predicting Y (*i.e.*, whether a term is a facet term) and Z (*i.e.*, whether two terms are in a same facet) independently. It first selects a set of candidate terms as facet terms by thresholding on $P(y_i|t_i)$, ignoring the dependences between y_i and its connected variables in Z. Then, QFI clusters only the selected terms into facets, using $P(z_{i,j}=0|p_{i,j}, y_i=1, y_j=1)$ as the distance measure. This algorithm has two tuning parameters, w_{min} for thresholding $P(y_i|t_i)$, and d_{max} for thresholding cluster diameter.

QFJ tries to perform joint inference on Y, Z by approximately maximizing their the posterior with respectively to Y and Z iteratively. It first guesses a set of candidate terms as initial facet terms. Then it clusters those selected terms using a greedy approach. After clustering, QFJ checks whether each term "fits" in its cluster based on the posterior, and removes outliers. The algorithm repeats the process (clustering and removing outliers) until convergence. QFJ has no tuning parameters, and the inference is purely guided by the posterior probability.

5. QUERY FACET EVALUATION MEASURES

In this section, we re-examine $PRF_{\alpha,\beta}$ measure proposed in our previous work [15], which will be used for evaluating query facet extraction under precision-oriented scenarios and developing our new approaches under the scenarios.

The idea of query facet evaluation is to compare extracted facets with ground truth facets constructed by human annotators. To help describe the measure, we will use superscript "$*$" to distinguish ground truth from system output. For example, y_i^* is a ground truth term label, while y_i is a term label predicted by the system. $T_{\mathcal{F}}^*$ is a set consisted by all the terms in the ground truth facets, while $T_{\mathcal{F}}$ is consisted by all terms in the system extracted facets.

The evaluation can be viewed from two perspectives. First, it can be evaluated as a classification task to measure how well models find facet terms. Second, it can be evaluated as a clustering task to measure how well models group facet terms together. $PRF_{\alpha,\beta}$ combines the classification and clustering evaluation together as described below.

Term Precision and Recall. The classification performance can be measured by term precision (*i.e.*, precision of the selected candidate terms being facet terms) and term recall (*i.e.*, recall of facet terms). They can be calculated as below, where subscript "c", "s", "g" stands for "correct", "system", "ground truth" respectively.

- Term precision: $TP = \frac{T_c}{T_s}$, where T_c is the number of correct facet term selected, T_s is the number of terms select by the system.
- Term recall: $TR = \frac{T_c}{T_g}$, where T_c is as defined above, T_g is the number of facet terms in the ground truth.
- Term F1: $TF = \frac{2T_c}{T_s+T_g}$ is the F1 combination (or the harmonic mean) of TP and TR.

Quantity T_c, T_s, T_g can be more precisely defined using term labels y_i and y_i^* as

$$T_c = \sum_i y_i y_i^*, \;\; T_s = \sum_i y_i, \;\; T_g = \sum_i y_i^*. \tag{4}$$

Term clustering. Our re-examination finds that the term clustering measure used in $PRF_{\alpha,\beta}$ could double-count term recall factor. The problem stems from that the terms being clustered by the model can be different from the terms clustered in the ground truth, *i.e.*, $T_{\mathcal{F}} \neq T_{\mathcal{F}}^*$. $T_{\mathcal{F}}$ may include wrong terms or miss correct facet terms. Standard clustering measures typically cannot handle these cases properly. Therefore, in our previous work [15], we adjust the extracted facets $\mathcal{F}$ as if only facet terms in $T_{\mathcal{F}}^*$ were clustered by the system. This is done by removing incorrect terms ($t \in T_{\mathcal{F}} - T_{\mathcal{F}}^*$) from $\mathcal{F}$, and adding each missing facet terms ($t^* \in T_{\mathcal{F}}^* - T_{\mathcal{F}}$) as singletons. The previous work claimed that by this adjusting, term clustering performance does not take into account the effectiveness of finding facet terms, but we now find it actually incorporates term recall factor. Analytically, we can see that when a system fails to find a facet term, by assuming it being a singleton, the clustering performance will be hurt (unless the facet term is a singleton in the ground truth). Empirically, we find systems return large sized facets when tuned on term clustering performance based on the adjusting. For example, on average, QFI returns 509.8 terms per query, while there is only 81.2 facet terms per query in the ground truth. Therefore, by combining term precision, recall and clustering performance, $PRF_{\alpha,\beta}$ actually double-counts the term recall factor by this adjusting when measuring clustering performance.

Since the mistakes of finding a wrong term and missing facet terms have already been accounted for in term precision and recall, we think term clustering performance should be measured only on the facet terms models selected correctly (*i.e.*, $T_{\mathcal{F}} \cap T_{\mathcal{F}}^*$) in $PRF_{\alpha,\beta}$. Thus, we instead adjust

clusters by removing incorrect terms ($t \in T_{\mathcal{F}} - T^*_{\mathcal{F}}$) in $\mathcal{F}$, as well as missing facet terms ($t^* \in T^*_{\mathcal{F}} - T_{\mathcal{F}}$) in $\mathcal{F}^*$. We find this overlap adjusting results in more reasonable returned facet size when tuned on clustering performance measures. The average number of facet terms returned per query by QFI now decreases to 157.07.

After cluster adjusting, facet term clustering performance is measured by pair-counting F1 measure. Here the pair-counting F1 measure treats term clustering as classification on whether each pairs of terms are in a same facet, and then combines pair precision and recall using F1 measure. Pair precision and recall can be calculated as below. (The subscripts carry the same meaning as in term precision and recall.)

- Pair precision: $PP = \frac{P_c}{P_s}$, where P_c is the number of term pairs the model clustered together that are indeed in a same facet in the ground truth, P_s is the number of term pairs the model clustered together.
- Pair recall: $PR = \frac{P_c}{P_g}$, where P_c is as defined above, T_g is the number of term pairs clustered together in the ground truth.
- Pair F1: $PF = \frac{2P_c}{P_s+P_g}$ is the F1 combination (or the harmonic mean) of PP and PR.

Quantity P_c, P_s, P_g can be more precisely defined using term labels y_i,y^*_i and pair labels $z_{i,j}$, $z^*_{i,j}$ as

$$P_c = \sum_{i,j} z_{i,j} z^*_{i,j},\ P_s = \sum_{i,j} z_{i,j} y^*_i y^*_j,\ P_g = \sum_{i,j} z^*_{i,j} y_i y_j, \quad (5)$$

where term labels y_i,y^*_i are used to perform the overlap adjusting as described previously.

Combining term precision, recall and clustering. The quality of query facet extraction is intrinsically multi-faceted. Different applications or scenarios might have different emphasis in the term precision, recall and clustering. To address this issue, $PRF_{\alpha,\beta}$ combines the three factors together, using weighted harmonic mean in a similar way as F-measures,

$$PRF_{\alpha,\beta}(TP, TR, PF) = \frac{(\alpha^2+\beta^2+1)}{\frac{\alpha^2}{TP} + \frac{\beta^2}{TR} + \frac{1}{PF}}, \quad (6)$$

where $\alpha, \beta \in [0, +\infty)$ are used to control the weight between the three factors in the same way as "β" in F-measures [26]. α and β can be interpreted as the importance of TP and TR compared to PF respectively. More formally, we have

$$\begin{aligned} &when\ \alpha = \frac{TP}{PF},\ \frac{\partial PRF_{\alpha,\beta}}{\partial TP} = \frac{\partial PRF_{\alpha,\beta}}{\partial PF} \\ &when\ \beta = \frac{TR}{PF},\ \frac{\partial PRF_{\alpha,\beta}}{\partial TR} = \frac{\partial PRF_{\alpha,\beta}}{\partial PF}. \end{aligned} \quad (7)$$

The intuition behind is that we want to specify the TP/PF ratio at which the user is willing to trade an increment in TP for an equal loss in PF, and similarly for TR/PF.

To evaluate query facet extraction under the precision-oriented scenario, we can set a high α and/or low β. For example, we can set $\alpha=2, \beta=1$ to evaluate the case where TP is twice important than TR and PF. Perhaps more reasonably, we can only down-weight the recall factor, by setting $\alpha=1, \beta=1/3$ to evaluate the case where TP and PF is three times important than TR.

To help develop our empirical utility maximization approach, we can rewrite $PRF_{\alpha,\beta}$ as a function of term and pair quantities $PRF_{\alpha,\beta}(T_c, T_s, T_g, P_c, P_s, P_g)$ as follows:

$$PRF_{\alpha,\beta} = \frac{2(\alpha^2+\beta^2+1)T_c T_p}{2\alpha^2 T_s P_c + 2\beta^2 T_g P_c + T_c P_s + T_c P_g}. \quad (8)$$

It is easy to see $PRF_{\alpha,\beta}$ can be also rewritten as a function of predicted labels and ground truth labels, $PRF_{\alpha,\beta}(Y, Z, Y^*, Z^*)$, by substituting term and pair quantities in Equation 8 using Equation 4 and 5.

6. EMPIRICAL UTILITY MAXIMIZATION

In this section, we will describe our empirical utility maximization (EUM) approach that directly optimize $PRF_{\alpha,\beta}$ for training QF model (described in Section 4).

The QF model can be viewed as a model which takes in candidate terms and term pairs and predicts their labels, $(Y, Z) = h(T_{\mathcal{L}}, P_{\mathcal{L}}; \lambda, \mu)$. The parameters λ, μ are trained by maximizing the conditional likelihood of the labels, $l(\lambda, \mu)$ as defined in Equation 3. One problem with the maximum likelihood estimation is the likelihood target can be loosely related to performance measure $PRF_{\alpha,\beta}$, especially in the precision-oriented scenario, where term recall are less important than other factors (as we will show in Section 8).

Therefore, we propose an alternative way of training the model $h(T_{\mathcal{F}}, P_{\mathcal{F}})$ by directly optimizing $PRF_{\alpha,\beta}$ measure. Our goal is to maximize the expected utility (or performance),

$$E_{\mathcal{P}}\big[PRF_{\alpha,\beta}(h(T_{\mathcal{L}}, P_{\mathcal{L}}), Y^*, Z^*)\big], \quad (9)$$

where $\mathcal{P}$ is the underlying and unknown distribution of our data $(T_{\mathcal{L}}, P_{\mathcal{L}}, Y^*, Z^*)$. In order to train the model, empirical utility maximization (or equivalently empirical risk minimization) is usually used, which tries to maximizes the above utility objective function over empirical data, $\mathcal{D} = \{T^{(i)}_{\mathcal{L}}, P^{(i)}_{\mathcal{L}}, Y^{*(i)}, Z^{*(i)} | i = 1...n\}$. The empirical utility is given below,

$$\begin{aligned} U(\lambda, \mu) &= E_{\mathcal{D}}\big[PRF_{\alpha,\beta}(h(T_{\mathcal{L}}, P_{\mathcal{L}}), Y^*, Z^*)\big] \\ &= \frac{1}{n}\sum_{i=1}^{n} PRF_{\alpha,\beta}(T^{(i)}_c, T^{(i)}_s, T^{(i)}_g, P^{(i)}_c, P^{(i)}_s, P^{(i)}_g), \end{aligned} \quad (10)$$

where we use the uniform distribution over empirical data to replace the unknown distribution, and replace the $PRF_{\alpha,\beta}$ term with $PRF_{\alpha,\beta}$ calculation based on term and pair quantities $PRF_{\alpha,\beta}(T_c, T_s, T_g, P_c, P_s, P_g)$ as defined in Equation 8.

Our goal now is to find $(\lambda, \mu) = \text{argmax}_{\lambda,\mu} U(\lambda, \mu)$. Unfortunately, this objective is difficult to optimize. The basic quantities involved are integers, and the optimization objective is a piecewise-constant function of the parameters λ, μ. The non-smoothness is due to that the dependent variable y_i and $z_{i,j}$ take only discrete values $\{0, 1\}$. For example, $U(\lambda, \mu)$ contains integer quantity $T_c = \sum_i y_i y^*_i$ that counts the correct facet terms labeled. According to QFI (see Section 4), y_i is predicted as either 1 or 0 by thresholding its term probability $P(y_i = 1|t_i)$ as:

$$y_i = 1\{P(y_i = 1|t_i) > w_{min}\}, \quad (11)$$

where $P(y_i = 1|t_i) = \frac{1}{1+\exp\{-\sum_k \lambda_k f_k(t_i)\}}$ (defined in Equation 1) involves parameter λ.

In generally, we can approximate discrete variables by their expectation to obtain a smooth objective function [11]. In our case, by assuming independence between all the labels, y_i can be approximated by its expectation as,

$$\widetilde{y}_i = E[y_i] = P(y_i = 1|t_i) = \sigma(\lambda^T f(t_i)), \quad (12)$$

where we use $\sigma(x) = \frac{1}{1+\exp\{-x\}}$ to denote the logistic function used in Equation 1, and use vector-representation for λ and feature $f(t_i)$ for convenience. Similarly, we approximate $z_{i,j}$ by its expectation assuming full independent condition as

$$\begin{aligned}\widetilde{z}_{i,j} &= E[z_{i,j}] = P(z_{i,j}=1, y_i=1, y_j=1|t_i, t_j, p_{i,j}) \\ &= P(z_{i,j}=1|p_i, y_i=1, y_j=1)P(y_i=1|t_i)P(y_j=1|t_j) \\ &= \sigma(\mu^T g(p_{i,j}))\sigma(\lambda^T f(t_i))\sigma(\lambda^T f(t_j)).\end{aligned} \tag{13}$$

In the same way, we can approximate term and pair quantities (*i.e.*, T_c, T_s, P_c, P_s, P_g) by their expectation. It is easy to see that, under the full independence assumption between all labels, their expectation can be obtained by substituting y_i and $z_{i,j}$ in Equation 4 and 5 with their expectation $E[y_i]$ and $E[z_{i,j}]$. For example, we can approximate $T_c \approx \widetilde{T}_c$ by

$$\widetilde{T}_c = E[T_c] = \sum_i E[y_i] y_i^* = \sum_i \sigma(\lambda^T f(t_i)) y_i^*. \tag{14}$$

Based on the approximated term and pair quantities, we can rewrite our optimization objective as

$$\widetilde{U}(\lambda, \mu) = \frac{1}{n}\sum_{i=1}^{n} PRF_{\alpha,\beta}(\widetilde{T}_c^{(i)}, \widetilde{T}_s^{(i)}, T_g^{(i)}, \widetilde{P}_c^{(i)}, \widetilde{P}_s^{(i)}, \widetilde{P}_g^{(i)}), \tag{15}$$

which can now be maximized numerically. More specially, we used gradient ascent for maximizing $\widetilde{U}(\lambda, \mu)$. The derivatives of $\widetilde{U}(\lambda, \mu)$ can be easily obtained based on the derivatives of $\widetilde{y}_i$, $\widetilde{z}_{i,j}$, as we given below,

$$\begin{aligned}\nabla_\lambda \widetilde{y}_i(\lambda) &= \sigma_i(1-\sigma_i)\lambda, \\ \nabla_\lambda \widetilde{z}_{i,j}(\lambda) &= \sigma_{i,j}\sigma_i\sigma_j(2-\sigma_i-\sigma_j)\lambda, \\ \nabla_\mu \widetilde{z}_{i,j}(\mu) &= \sigma_i\sigma_j\sigma_{i,j}(1-\sigma_{i,j})\mu,\end{aligned} \tag{16}$$

where $\sigma_i \equiv \sigma(\lambda^T f(t_i))$, $\sigma_{i,j} \equiv \sigma(\mu^T g(p_{i,j}))$. Note that the function $\widetilde{U}(\lambda, \mu)$ is generally not concave. We can deal with this problem by taking the maximum across several runs of the optimization algorithm starting from random initial values. After training, we use the original inference QFI and QFJ as describe in Section 4 to predict labels and induce facets.

7. SELECTIVE QUERY FACETING

In this section we describe selective query faceting – our selective method for query facet extraction. The idea is motived by the variance in extraction performance we observed – depending on the nature of queries and extraction models, quality of the extracted facets varies drastically from excellent to poor and complete noise. For example, queries about products, such as "toilet" and "volvo", tend to have more high-quality candidate facets extracted and are therefore easier than other complex queries, such as "self motivation", to find query facets. In our experiments, we also find $PRF_{\alpha,\beta}$ could range from 0 to above 0.8.

Selective Query Faceting. Similar as the idea of selective query expansion [30, 1] we can selectively present facets to users based on the extraction performance of each queries. Ideally, we can only show facets for good performing queries and avoid bad ones to improve user satisfaction, as in the precision-oriented scenario, it may be more desirable to leave users with a clean keyword-search interface than showing poor-quality facets. To support this selective query faceting, a key problem need to be addressed is the prediction of extraction performance. We find a simple score based on the expectation of $PRF_{\alpha,\beta}$ could predict extraction performance fairly well. Next, we will describe our extraction performance prediction method.

Performance Prediction. In performance prediction, our goal is to predict the extraction performance for a given query with its extracted facets. We focus on predicting $PRF_{\alpha,\beta}$, and leave prediction of other measures as future work. The prediction could be done by using single indicator scores (like the clarity score in prediction retrieval performance [4]), or by combining different features using regression or classification models. No matter which approach, we first need to find good indicators/features for estimating the performance.

To find effective features, a natural way is to investigate the probabilistic model we have already learned in the QF method, because the learned probabilities already incorporate beliefs about the correctness of corresponding outputs. For example, we can use the term probability defined in Equation 1 to estimates the "likelihood" of the output terms are indeed facet terms, and use the pairs probability defined in Equation 2 to estimated the "likelihood" of the term pairs in a same extracted facets indeed belong to a query facet.

In order to use the term and pair probabilities as features, we need to aggregate them in some ways, because these probabilities are for terms and pairs, not directly for whole extracted facet set. We investigates two ways of aggregation. First, from the perspective of data fitness, we can directly use log-likelihood of extracted facets to measure the fitness. For example, we can use the whole log-likelihood based on Equation 3, and we can also use the log-likelihood for only the terms or only the pairs based on the first term and second terms in the equation respectively.

Second, from the perspective of directly estimating utility (performance), we can aggregate the probabilities for estimating $PRF_{\alpha,\beta}$ directly in a similar way as our empirical utility maximization approach. More specially, we can estimate $PRF_{\alpha,\beta}$ performance based on the expected term and pair quantities under the learned model. The estimates can be obtained as follows,

$$\begin{aligned}\widehat{TP} &= \frac{\sum_i P(t_i) y_i}{\sum_i y_i}, \ \widehat{TR} = \frac{\sum_i P(t_i) y_i}{\sum_i P(t_i)}, \\ \widehat{PT} &= \frac{\sum_{i,j} P(p_{i,j}) z_{i,j}}{\sum_{i,j} z_{i,j}}, \ \widehat{PR} = \frac{\sum_{i,j} P(p_{i,j}) z_{i,j}}{\sum_{i,j} P(p_{i,j}) y_i y_j},\end{aligned} \tag{17}$$

where we use $P(t_i) \equiv P(y_i = 1|t_i)$, $P(p_{i,j}) \equiv P(z_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1)$ for simplification. Estimates of TF, PF and can be easily obtained by substitute TP, TR, PP, PR with their estimates in the corresponding equations in Section 5. Estimate of $PRF_{\alpha,\beta}$ can be obtained by substituting TP, TR and PF with their estimates in Equation 6. We call this estimate of $PRF_{\alpha,\beta}$ "PRF score".

To investigate the effectiveness of the two types of features, we analyze the correlation between extraction performance and each individual features. First, we find that PRF score has strong correlation (0.6249 with p-value 3.6×10^{-12}) with the performance $PRF_{\alpha=1,\beta=1}$. This suggest 1) PRF score is a good indicator for extraction performance, and might be effective in performance prediction, and 2) our

estimation of $PRF_{\alpha=1,\beta=1}$ based on its expectation is effective. Second, we find utility-based features PRF scores, $\widehat{TF}$, $\widehat{TR}$ correlate better with $PRF_{\alpha,\beta}$ performance than other likelihood-based features. This validates our assumption that likelihood can be loosely related to the performance measure, and utility could be a better optimization objective. (We omit detail results due to space limitation.)

We combine the proposed features in linear regression and logistic regression models. However, we find the results are not significantly better than simply using PRF score for prediction, which could be caused by the linear dependence between those features. Thus, we propose to use only *PRF* score for extraction performance prediction, which is simple and effective as we will show in Section 8. We also test other features based on statistic aggregates of the term and pair probabilities, including minimum, maximum, mean, sum and standard deviation. However, they show relatively low correlation with $PRF_{\alpha,\beta}$, and thus we do not report the results here due to space limitation.

After choosing PRF score as the performance predictor, selective query faceting can be easily done by thresholding this score to decide to show or avoid showing query facet results for each query. We carry out experiments to evaluate its effectiveness in Section 8.

8. EXPERIMENTS

Our experiments aim to investigate mainly three research questions. First, we want to test whether existing query facet extraction methods adapt to precision-oriented scenarios. Second, we want to test if our empirical utility maximization approach is effective in precision-oriented scenarios. Last, we want to test whether the PRF score effectively predict extraction performance, and support selective query faceting. We will first describe our experimental settings, then present experiment results for each of the research questions.

8.1 Experimental Settings

Data. We use two datasets from previous work, which we name QF13 [15] and FWS14 [16]. QF13 contains 100 web search queries, their top 100 web search results from a commercial web search engine, and their query facet annotation. FWS14 contains 196 queries from TREC Web Track and their query facet annotation. We perform the same retrieve model to obtain top 100 search results for these queries as in the original work. The query facet annotation was done by pooling facet results from different models, and then having the pooled terms re-grouped by human annotators into query facets [15]. Our facet candidates are extracted as described in Section 3.

Evaluation. We use $PRF_{\alpha,\beta}$ as the evaluation measures, as well as term precision (*i.e.*, *TP*), term recall (*i.e.*, *TR*), and term clustering F1 (*i.e.*, *PF*). We choose this measure because it has the flexibility of adjusting emphasis between "facet precision" and "facet recall", which naturally suits well with the precision-oriented problem. When $\alpha=\beta=1$, $PRF_{\alpha=1,\beta=1}$ is used to evaluate the case where term precision, term recall and term clustering are equally important. To evaluate facets under precision-oriented scenarios, we set high $\alpha \in \{2,3,\ldots,10\}$ with fixed $\beta=1$. The settings correspond to the cases where term precision is twice to ten times important as both term recall and term clustering. Without any prior knowledge, it is more fair to assume that term precision and clustering are equally important (they are both "precision" factors for query facets), therefore we will focus more on only down-weight term recall by setting $\beta \in \{\frac{1}{2},\frac{1}{3},...,\frac{1}{10}\}$ or equivalently $\frac{1}{\beta} \in \{2,3,\ldots,10\}$ with fixed $\alpha=1$. The settings correspond to the case where term precision and clustering is twice to ten times importance as term recall. Following the previous work [15, 16], we evaluate top 10 facets returned from each models.

We use 10-fold cross validation on QF13 and 4-fold cross validation on FWS14, as in the original work, for training, tuning (if applicable) and testing models. Models are tuned on the same $PRF_{\alpha,\beta}$ measure that they are tested on. Unless specified else, statistically significant test is performed by using paired t-test with 0.05 as the p-value threshold.

Methods. We study five query facet extraction models briefly summarized as below.

- pLSA and LDA [32, 15]: pLSA and LDA was applied on candidate facets for facet refining. After training, the topics are returned as query facets, by using top terms in each topic. We tune the number of facets and number of facet terms in each facets.
- QDM [7]: this is an unsupervised clustering method that applies a variation of the Quality Threshold clustering algorithm [8] to cluster the candidate facets with bias towards important ones. We tune the diameter threshold, weight threshold for valid cluster and the threshold for selecting terms.
- QFI and QFJ [15]: the two models have been described in Section 4. They are reported to be the best across several evaluation measures [15, 16], therefore we primarily focus on them. Except for the difference of independent and joint inference, the two models are different in that QFI has tuning parameters that can be tuned for given measures, while QFJ do not, as it tries to optimize log-likelihoods. For QFI, we tune the weight threshold for facet terms w_{min}, and the diameter threshold d_{max}. There are no tuning parameters for QFJ.

We study two ways of training the graphical model (see Section 4) for QFI and QFJ.

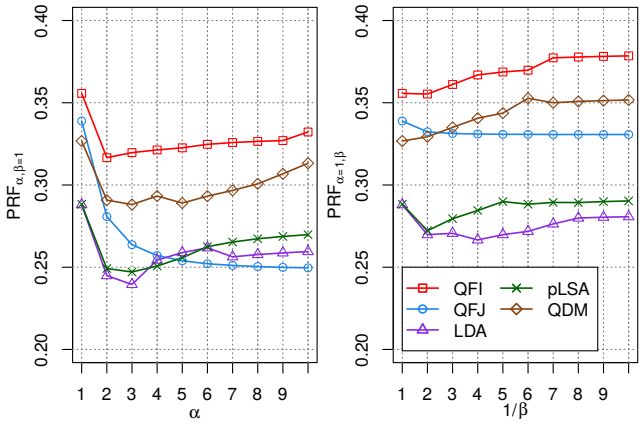
- MLE: our previous work [15] used likelihood as the optimization objective and performed maximum likelihood estimating for training (see Section 4).
- EUM: since likelihood can loosely related to the performance measure, we propose to use empirical utility maximization to directly optimize the $PRF_{\alpha,\beta}$ measure during training (see Section 6). With different α,β, we can use different versions of $PRF_{\alpha,\beta}$ as the optimization objective. We test three runs by setting ($\alpha=1,\beta=1$), ($\alpha=2,\beta=1$), ($\alpha=1,\beta=\frac{1}{2}$). We denote the different runs by add α,β subscript in "EUM" (*e.g.*, $EUM_{2,1}$ stands for EMU training using $PRF_{\alpha=2,\beta=1}$ as the optimization objective).

8.2 Evaluation under Precision-Oriented Scenarios

We first investigate if the five existing models can adapt to precision-oriented scenarios by evaluation based on $PRF_{\alpha,\beta}$ with different α,β settings. In Figure 2, we show $PRF_{\alpha,\beta}$ performance of different α (*i.e.*, term precision is more important than term recall and clustering) on the left, and of

different β (*i.e.*, term precision and clustering are more important than term recall) on the right. We test all the five models with QFI, QFJ trained by MLE. We report results on FWS14 (observations are similar for QF13).

Figure 2: $PRF_{\alpha,\beta}$ performance with different α (left, fixed $\beta=1$) and different β (fixed $\alpha=1$, right) settings for existing methods on FWS14.



(a) $PRF_{\alpha,\beta=1}$ **(b)** $PRF_{\alpha=1,\beta}$

First, we find QFJ does not adapt well to precision-oriented scenarios. From the figure, we can see the superiority of QFJ over other models becomes less evident (on QF13) or disappears on FWS14, when moving from the normal case to precision-oriented cases. This is due to that QFJ tries to optimize log-likelihood for inferencing, and it cannot be tuned on the performance measures like other models. So it returns the same results for the normal case and precision-oriented scenarios. Second, we find generally QFI and QDM can adapt better than the other models to the precision-oriented scenarios, with QFI consistently better than all the other models on both datasets. The adaptability of the two models can be explained by their tuning procedure. For example, depending on the target performance measure, QFI can set different threshold w_t for select facet terms. Overall, we find QFI (under MLE training) is the best among these existing models for the normal case, as well as precision-oriented cases.

To further analyze how QFI adapt to the precision-oriented scenarios, in Table 2, we report $PRF_{\alpha,\beta}$ together with *TP*, *TR*, *PF* and facet size (the total number of terms returned for a query) when setting different β in $PRF_{\alpha,\beta}$. From the table, we find that as term recall factor becomes less and less important (or equivalently as the precision factors becomes more and more important), QFI becomes more and more conservative in selecting terms. The number of terms returned on average for each queries ("size" in the table) decreases from 89.5 to 45.2. Term precision *TP* thus increases significantly, while term recall *TR* and term clustering *PF* decreases. This indicates, by tuning on the performance measure, QFI tries to find a good balance between the tree factors for each scenarios.

8.3 EUM Performance

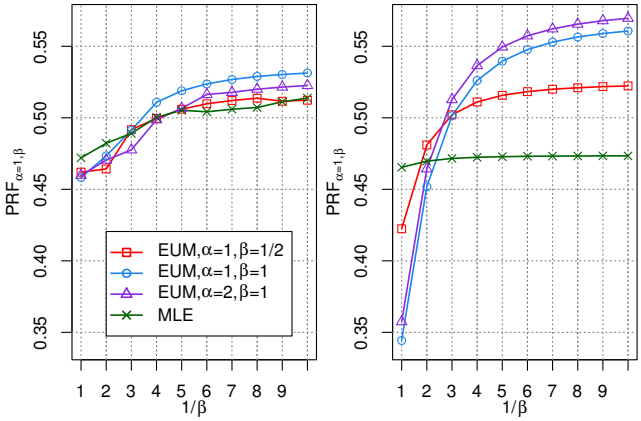
Next, we compare EUM and MLE training to test the effectiveness of the EUM approach we proposed. We first compare EUM and MLE training using both QFI and QFJ

Table 2: $PRF_{\alpha,\beta}$ performance with its *TP*, *TR*, *PF* under different β settings (fixed $\alpha=1$) for QFI on QF13. "Size" reports the average number of terms returned for each queries.

$\frac{1}{\beta}$	$PRF_{\alpha,\beta}$	TP	TR	PF	Size
1	0.4720	0.4450	0.4881	0.6209	89.5
2	0.4822	0.4896	0.4186	0.6192	70.7
3	0.4891	0.5108	0.3574	0.5989	56.5
4	0.5003	0.5291	0.3498	0.5925	53.0
5	0.5053	0.5348	0.3306	0.5928	48.9
6	0.5042	0.5343	0.3194	0.5834	47.1
7	0.5060	0.5343	0.3194	0.5834	47.1
8	0.5072	0.5343	0.3194	0.5834	47.1
9	0.5112	0.5364	0.3172	0.5864	46.7
10	0.5138	0.5365	0.3097	0.5824	45.2

in Figure 3. Due to space limitation we only report results for $PRF_{\alpha=1,\beta}$ (*i.e.*, fixed $\alpha=1$ with different β settings) on QF13. Observations are similar for other cases. The figure shows QFI with EUM and MLE training on the left, and QFJ results on the right. We report three runs of EUM, which use $PRF_{\alpha,\beta}$ under different α,β settings (specified in the legend) as the training target.

Figure 3: $PRF_{\alpha=1,\beta}$ performance for MLE and EUM training using QFI (left) and QFJ (right) on QF13. The three EUM runs use $PRF_{\alpha,\beta}$ under different α,β settings (specified in the legend) as the training target.



(a) QFI **(b) QFJ**

From Figure 3, for QFI, we find there are no statistically significant difference between MLE and EUM in most cases, even though generally EUM obtains slightly better $PRF_{\alpha,\beta}$ than MLE. This can be explained by that QFI under MLE has already incorporated the $PRF_{\alpha,\beta}$ learning target due to that it is tuned on $PRF_{\alpha,\beta}$. Essentially, we can view QFI (under MLE training) as a model that is trained on likelihood to find a small tuning space to enable optimization on given performance measures by hand tuning.

Differently, for QFJ, we find EUM can improve over MLE largely under the precision-oriented scenarios. The difference between EUM and MLE are statistically significant for all $1/\beta > 2$ and for all the three EUM runs. This indicates 1) utility (performance measure) is a better optimization objective than likelihood and 2) our approximation of $PRF_{\alpha,\beta}$ based on its expectation is effective.

To study how EUM training affects QFJ in more details, as an example, we show $PRF_{\alpha=1,\beta=0.1}$ together with its *TP*, *TR*, *PF* and facet size in Table 3.

Table 3: $PRF_{\alpha=1,\beta=0.1}$ with *TP*, *TR*, *PF* for MLE and EUM training on QF13. Subscripts of EUM indicates the α, β setting used for its optimization target $PRF_{\alpha,\beta}$.

model	Training	$PRF_{\alpha,\beta}$	TP	TR	PF	Size
QFJ	MLE	0.4734	0.3986	0.4832	0.6961	97.0
QFJ	$EUM_{1,1}$	0.5223	0.4884	0.3341	0.6702	54.8
QFJ	$EUM_{2,1}$	0.5696	0.5711	0.2328	0.6705	33.9
QFJ	$EUM_{1,0.5}$	0.5607	0.5710	0.2229	0.6620	33.0

From Table 3, first, we can see when trained on EUM under precision-oriented settings (*i.e.*,$EUM_{2,1}$ and $EUM_{1,0.5}$), QFJ are more conservative in selecting terms than in MLE training. When moving from MLE to EUM training, its facet size becomes much smaller (*i.e.*, 97 to 33), *TP* increases largely while *TR* decreases largely. This effect is desirable under the precision-oriented scenarios, in which we care much more about precision than recall, as reflected by the improvement in $PRF_{\alpha=1,\beta=0.1}$ shown in the table.

Second, by comparing $EUM_{1,1}$ with $EUM_{2,1}$, $EUM_{1,0.5}$ in Table 3, we can see $EUM_{2,1}$, $EUM_{1,0.5}$ trained models behave more conservatively than $EUM_{1,1}$ trained models. This suggest our training is effective – as we change the training target $PRF_{\alpha,\beta}$ parameter from ($\alpha = 1, \beta = 1$) to ($\alpha = 2, \beta = 1$) and ($\alpha = 1, \beta = 0.5$)), it learns that we are putting more emphasis on precision, and thus behaves more conservatively.

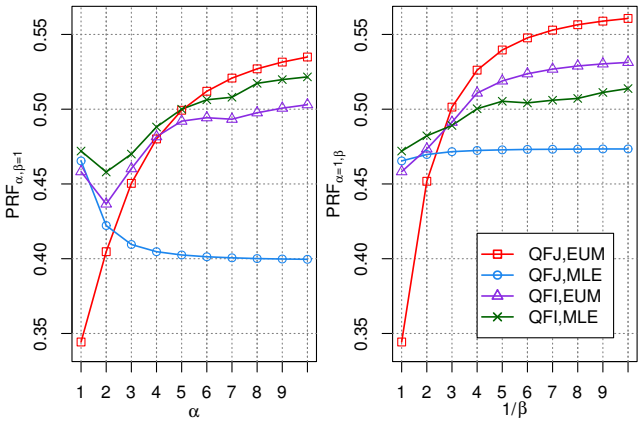
Last, the improvement of QFJ in precision oriented scenarios raises a question – will it outperform previous best model QFI under precision-oriented scenarios? We test this in Figure 4. In the figure, we compare QFJ under EUM training with other baselines, including QFI under MLE (representing the state-of-the-art baseline) and EUM training. We only report results under $EUM_{1,0.5}$ training on QF13 (results are similar in other cases). From the figure we find QFJ under EUM training outperforms other models in the precision-oriented scenarios. The difference between QFJ,EUM and the state-of-the-art method QFI,MLE are statistically significant for $PRF_{\alpha=1,\beta}$ when $\frac{1}{\beta} > 4$.

8.4 Extraction Performance Prediction

To predict query facet extraction performance, we build linear regression models using only PRF score (see Section 7) as the feature (with intercept). We test the models for predicting $PRF_{\alpha,\beta}$ under different α, β, based on 10-fold cross validation on QF13 for QFI in Table 4. We report root-mean-square deviation (RMSD), Pearson correlation (R), and p-values for the significance of correlation.

The results in Table 4 show fairly strong RMSD values and strong positive correlations between the predicted $PRF_{\alpha,\beta}$ and real $PRF_{\alpha,\beta}$ performance for most cases. For example, p-value 1.4×10^{-11} for the first row indicates that it is extremely unlike that the predicted $PRF_{\alpha=1,\beta=1}$ performance has no relationship with the actually performance. We also see one exception. For $PRF_{\alpha=5,\beta=1}$ we only see fair correlation, which may due to that we use $\alpha = 1, \beta = 1$ for computing our PRF score, while in $PRF_{\alpha=5,\beta=1}$ the three factors are more unbalanced weighted.

Figure 4: $PRF_{\alpha,\beta}$ performance with different α, β settings for QFI and QFJ under MLE and EUM training on QF13. $EUM_{1,0.5}$ run result is reported for EUM.



(a) $PRF_{\alpha,\beta=1}$ **(b)** $PRF_{\alpha=1,\beta}$

Table 4: Linear regression results based on 10-fold cross-validation for predicting $PRF_{\alpha,\beta}$ performance. RMSD – root-mean-square deviation, R – Pearson correlation.

Measure	RMSD	R	p-value
$PRF_{\alpha=1,\beta=1}$	0.1110	0.6112	1.4×10^{-11}
$PRF_{\alpha=1,\beta=0.2}$	0.1800	0.5745	4.1×10^{-10}
$PRF_{\alpha=1,\beta=0.1}$	0.1882	0.5566	1.8×10^{-9}
$PRF_{\alpha=5,\beta=1}$	0.2109	0.2958	0.0028
$PRF_{\alpha=10,\beta=1}$	0.2245	0.4028	3.2×10^{-5}

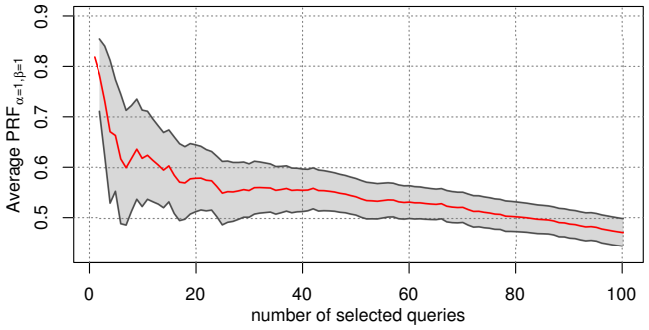
8.5 Evaluating Selective Query Faceting

Next, we study the effectiveness of selective query faceting based on the predicted score. Recall that our selective method is done by thresholding the predicted performance for deciding show or avoid showing facets for each query (see Section 7). With a higher threshold, selective query faceting would select less queries to show facets, but users should obtain better performance for the facets that are presented to them. On the contrary, a lower threshold will result in selecting more queries to show facets, but the performance for the selected queries may be worse. There is a trade-off between the performance of selected queries and the coverage on queries for query faceting.

To evaluate selective query faceting, we plot the average $PRF_{\alpha,\beta}$ performance for queries selected by PRF score, when using different thresholds in Figure 5. The x-axis indicates the number of selected queries, while the y-axis indicates the average $PRF_{\alpha,\beta}$ performance for those selected queries. In addition to average $PRF_{\alpha,\beta}$, we also plot the standard error with 95% confidence intervals by the gray area (except for the case where only one query is selected). We report results on QF13 for a QFI run that are trained under MLE and evaluated on $PRF_{\alpha=1,\beta=1}$.

From Figure 5, we can see as we select less and less queries for presenting facets, generally the average performance for the selected queries increases. This indicates the query faceting method is fairly effective in selecting good performing queries and avoid bad ones. When 20 queries are selected, we ob-

Figure 5: Average $PRF_{\alpha,\beta}$ performance for selected queries. The gray area indicates standard error with 95% confidence intervals. Run: $PRF_{\alpha=1,\beta=1}$ as the measure with MLE trained QFI as the extraction model



tain 0.5792 $PRF_{\alpha=1,\beta=1}$ for the selected queries, comparing to 0.4720 when the selective method is not performed (*i.e.*, showing facets for all queries). The difference are statistically significant according to two-tailed two-sample t-test (p-value = 0.0034).

9. CONCLUSIONS AND FUTURE WORK

In this work, we study and improve query facet extraction under precision-oriented scenarios, which could help this technique to be used practically. We find the performance expectation can be used as an approximation to directly optimize the performance measure, which significantly improves existing models under precision-oriented scenarios. We propose PRF score based on the expectation of $PRF_{\alpha,\beta}$ to predict extraction performance. We show this score has fairly good prediction performance which enables selective query faceting that selects good performing queries to show facets, and improve the average extraction performance.

As a start for making query facet extraction more practical, this work only focuses on precision-oriented scenarios and only on the $PRF_{\alpha,\beta}$ performance measure. However, it also opens up several interesting directions for future work. First, it may be interesting to study recall-oriented scenarios, as high recall may be more desirable in some cases where users want to explore more (*e.g.*, exploratory search). Second, it would also be interesting to generalize the empirical utility maximization approach to optimize other performance measures (*e.g.*, facet ranking measure) for query facet extraction.

10. ACKNOWLEDGEMENT

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

11. REFERENCES

[1] G. Amati, C. Carpineto, G. Romano, and F. U. Bordoni. Query difficulty, robustness, and selective application of query expansion. In *ECIR*, volume 4, 2004.

[2] N. Balasubramanian, G. Kumaran, and V. R. Carvalho. Predicting query performance on the web. In *Proc. of SIGIR*, 2010.

[3] D. Cossock and T. Zhang. Subset ranking using regression. In *Learning theory*. Springer, 2006.

[4] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of SIGIR*, 2002.

[5] W. Dakka and P. G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *Proc. of ICDE*, 2008.

[6] H. M. de Almeida, M. A. Gonçalves, M. Cristo, and P. Calado. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *Proc. SIGIR*, 2007.

[7] Z. Dou, S. Hu, Y. Luo, R. Song, and J.-R. Wen. Finding dimensions for queries. In *Proc. of CIKM*, 2011.

[8] L. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome research*, 9(11), 1999.

[9] S. Hu, Z. Dou, X. Wang, T. Sakai, and J.-R. Wen. Search result diversification based on hierarchical intents. In *Proc. of CIKM*, 2015.

[10] S. Hu, Z.-C. Dou, X.-J. Wang, and J.-R. Wen. Search result diversification based on query facets. *Journal of Computer Science and Technology*, 30(4), 2015.

[11] M. Jansche. Maximum expected F-measure training of logistic regression models. In *Proc. of HLT/EMNLP*, 2005.

[12] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proc. of SIGIR*, 2000.

[13] T. Joachims. A support vector method for multivariate performance measures. In *Proc. of ICML*, 2005.

[14] C. Kohlschütter, P.-A. Chirita, and W. Nejdl. Using link analysis to identify aspects in faceted web search. In *SIGIR Workshop on Faceted Search*, 2006.

[15] W. Kong and J. Allan. Extracting query facets from search results. In *Proc. of SIGIR*, 2013.

[16] W. Kong and J. Allan. Extending faceted search to the general web. In *Proc. of CIKM*, 2014.

[17] K.-L. Kwok, L. Grunfeld, H. Sun, P. Deng, and N. Dinstl. TREC 2004 robust track experiments using PIRCS. In *TREC*, 2004.

[18] K. Latha, K. R. Veni, and R. Rajaram. AFGF: An automatic facet generation framework for document retrieval. In *Proc. of ACE*, 2010.

[19] C. Li, N. Yan, S. B. Roy, L. Lisham, and G. Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proc. of WWW*, 2010.

[20] D. R. Musicant, V. Kumar, A. Ozgur, et al. Optimizing F-measure with support vector machines. In *FLAIRS Conference*, 2003.

[21] E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for RDF data. In *Proc. of ISWC*, 2006.

[22] C. Quoc and V. Le. Learning to rank with nonsmooth cost functions. *Proc. of NIPS*, 19, 2007.

[23] E. Stoica and M. A. Hearst. Automating creation of hierarchical faceted metadata structures. In *Proc. of NAACL-HLT*, 2007.

[24] J. Teevan, S. Dumais, and Z. Gutt. Challenges for supporting faceted search in large, heterogeneous corpora like the web. *Proc. of HCIR*, 2008.

[25] S. Tomlinson. Robust, web and terabyte retrieval with Hummingbird SearchServer^{TM} at TREC 2004. In *Proc. of TREC*, 2004.

[26] C. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[27] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proc. of SIGIR*, 2007.

[28] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma. Directly optimizing evaluation measures in learning to rank. In *Proc. of SIGIR*, 2008.

[29] N. Ye, K. M. A. Chai, W. S. Lee, and H. L. Chieu. Optimizing f-measures: a tale of two approaches. In *Proc. ICML*, 2012.

[30] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proc. of SIGIR*, 2005.

[31] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proc. of SIGIR*, 2007.

[32] H. Zhang, M. Zhu, S. Shi, and J.-R. Wen. Employing topic models for pattern-based semantic class discovery. In *Proc. of ACL-IJCNLP*, 2009.

[33] Y. Zhou and W. B. Croft. Ranking robustness: a novel framework to predict query performance. In *Proc. of CIKM'06*.