Probabilistic Latent Query Analysis for Combining Multiple Retrieval Sources

Rong Yan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213
yanrong@cs.cmu.edu

Alexander G. Hauptmann
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213
alex+@cs.cmu.edu

ABSTRACT

Combining the output from multiple retrieval sources over the same document collection is of great importance to a number of retrieval tasks such as multimedia retrieval, web retrieval and meta-search. To merge retrieval sources adaptively according to query topics, we propose a series of new approaches called probabilistic latent query analysis (pLQA), which can associate non-identical combination weights with latent classes underlying the query space. Compared with previous query independent and query-class based combination methods, the proposed approaches have the advantage of being able to discover latent query classes automatically without using prior human knowledge, to assign one query to a mixture of query classes, and to determine the number of query classes under a model selection principle. Experimental results on two retrieval tasks, i.e., multimedia retrieval and meta-search, demonstrate that the proposed methods can uncover sensible latent classes from training data, and can achieve considerable performance gains.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval models

General Terms: Algorithms, Performance, Theory

Keywords: Retrieval, Query Class, Fusion, Learning

1. INTRODUCTION

A number of information retrieval tasks utilize retrieval outputs provided by multiple knowledge sources for the same document collection. For example, web retrieval [7] considers a single web page as a document retrievable based on several types of information, such as titles, anchor texts, main body texts as well as its linking relation to other pages. Multimedia retrieval [20, 1, 8] attempts to retrieve documents containing information extracted from multiple modalities, such as the speech transcription text, the low-level visual features of the images and a set of semantic concepts automatically detected in the video clip. Meta-search [15, 10, 11] starts with the ranked lists generated from different search engines on a single document collection. In all scenarios, retrieval systems usually benefit from combining various retrieval sources by exploiting complementary information from them.

Retrieval source combination has been examined by a significant body of previous work. Most current approaches fall into the category of *query-independent* methods, which adopt the same combination strategy for every query. Well-known examples are the CombSUM/CombMNZ methods [15] in meta-search. However, these methods might have limited effectiveness because optimal combination strategies often vary considerably for different query topics. To address this, recent work has led to *query-class* dependent combination approaches, which map the query space into one of the predefined query classes and learn the best combination strategy for each class from training data. For example, in order to improve web document retrieval, Kang et al. [7] classified queries into three categories where each category had its specific combination strategy. In multimedia retrieval, query-class combination methods using manually defined query classes [20, 1, 8] have been superior to query-independent combination in a number of standard video search evaluations [16]. Voorhees et al. [17] proposed a query clustering method for distributed IR, which groups the queries based on the number of common documents retrieved and creates centroids by averaging the query vectors inside the clusters. Recent advances in query difficulty classification [21] can also be viewed as quite similar. In these studies, each query is first classified into one of two groups based on retrieval "difficulty" and then coupled with corresponding merging schemes based on classification results.

Despite recent successes, query-class combination methods still have plenty of room for improvement. One major issue is that query classes usually need to be defined using expert domain knowledge. This manual design can work well when only a few query classes are needed, but it will become difficult for tens or hundreds of query classes, where each query in a class has to share similar characteristics and thus a similar combination strategy. If the query classes are not carefully defined, a large learning error from both the query-class assignment and the combination parameter estimation might result. Furthermore, current query-class methods do not allow mixtures of query classes, but at times such a mixture treatment could be helpful. For instance, the query "finding Bill Clinton in front of US flags" should be associated with both a "Person" class and a "Named Object" class rather than only one of these. Finally, determining the number of query classes remains an unanswered problem in these methods due to the nature of manual design. Some previous approaches [17, 8] can discover query classes by using clustering techniques. However, these approaches typically separate the processes of query class categorization and combination optimization into two sequential steps without being jointly optimized. Moreover, it is not straightforward for general clustering methods to handle mixtures of query classes in a principled manner.

Based on these considerations, it is desirable to develop a data-driven probabilistic combination approach that allows query classes and their corresponding combination parameters to be automatically discovered from the training data itself, rather than handcrafted using human knowledge. Therefore, we propose a new combination approach called probabilistic latent query analysis (pLQA) to merge multiple retrieval sources based on statistical latent-class models. The proposed approaches have advantages over query-independent and query-class combination methods in several ways: (1) they unify combination weight optimization and query-class categorization into a single discriminative learning framework; (2) they are able to automatically discover the latent query classes directly from training data; (3) they can handle mixtures of query classes in one query and (4) they can determine the number of query classes with an statistical model selection principle. Experiments are conducted on two retrieval applications, i.e., multimedia retrieval on the TRECVID'02-'05 collections [16] and meta-search on the TREC-8 collection [18]. The results show that the proposed approaches can uncover sensible latent classes from training data, and also demonstrate higher effectiveness in combining multiple retrieval sources.

2. A DISCRIMINATIVE FRAMEWORK FOR RETRIEVAL SOURCE COMBINATION

In this section, we present a discriminative combination framework that all of our proposed approaches are built on. Let us begin by introducing the basic notations and terminologies used in this work. The term *document* refers to the basic unit of retrieval throughout this paper. A query collection $\mathcal{Q}$ contains a set of queries $\{q_1, ..., q_t, ..., q_{M_Q}\}$ where q_t can have either a set of keywords, a detailed text descriptions and possibly image, audio, and video query examples. A search collection $\mathcal{D}$ contains a set of documents $\{d_1, ..., d_j, ..., d_{M_D}\}$. Let $y \in \{-1, 1\}$ indicate if document D is relevant or irrelevant to query Q. For each query q and document d, we can generate a bag of ranking features from N retrieval sources, denoted as $f_i(d, q)$. Our goal is to generate an improved ranked list by combining $f_i(d, q)$.

Conventional relevance-based probabilistic models [4] rank documents by sorting the conditional probability that each document would be judged relevant to the given query, i.e., $P(y = 1|D, Q)$ or denoted as $P(y_+|D, Q)$. Most well-known text retrieval models, such as the BIR model [13], proceed by inverting the position of y and D based on the Bayes rule and estimating the generative probabilities of document D in the relevant and irrelevant documents. However, the underlying model assumptions of these approaches such as the term independency could be invalid in practice. In contrast, discriminative models can directly model the classification boundary and typically make fewer model assumptions. They have been applied in many domains of text processing such as text classification and information extraction. Moreover, different retrieval sources usually provide heterogeneous types of outputs for combination including both query-dependent features and query-independent features. Generative models might face difficulties in the manual design of different model distributions for these outputs. Nallapati [12] has shown that in presence of heterogenous features, discriminative models are superior to generative models in a home-page finding task. Moreover, since the number of retrieval sources is usually much smaller than the number of text keywords, it allows a discriminative model to estimate the parameters more robustly given the same amount of the training data.

Taking these factors into account, we decided to utilize discriminative models to combine multiple retrieval sources. Formally, we model the posterior probability of the relevance as a logistic function on a linear combination of ranking features, i.e.,

$$P(y_+|D, Q) = \sigma\left(\sum_{i=0}^{N} \lambda_i f_i(D, Q)\right), \qquad (1)$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the standard logistic function and λ is the combination parameter for the output of i^{th} ranking features $f_i(D, Q)$. This logistic regression model presented in Eqn(1), a.k.a. the maximum entropy model, summarizes our basic retrieval source combination framework. It naturally provides a probabilistic interpretation for the retrieval outputs. Once the parameters are estimated, documents can be presented to users in descending order of $P(y_+|D, Q)$, or equivalently by the weighted sum of retrieval outputs $\sum_{i=0}^{N} \lambda_i f_i(D, Q)$. The optimization problem can be solved using Newton's method described in [2].

3. PROBABILISTIC LATENT QUERY ANALYSIS

The aforementioned discriminative retrieval framework provides a principled platform to support the task of retrieval source combination. However, in its current form, the model still has a problem in that the combination parameters λ_i are completely independent of the queries. In other words, the model will associate every possible query with the same set of combination parameters no matter what types of information needs users are expressing. To improve upon this, we study a more flexible combination approach called *probabilistic latent query analysis*(pLQA). It aims to determine the combination weights based on the latent mixing structure of the query space that can be automatically discovered under a statistical latent class model. In the rest of this section, we first discuss the basic form of the pLQA method and its parameter estimation strategies. To deal with unseen queries outside the training collection, we then extend pLQA to its adaptive version and kernel version.

3.1 Basic pLQA

It would be ideal if we could learn specific combination parameters for every possible query. However, given the virtually infinite number of query topics, it is impractical to learn the combination weights on a per query basis because we cannot collect enough training data individually. We need to come up with a trade-off to balance the difficulties of collecting training data and the ability to capture the idiosyncracy of the query space. To achieve this, we make the

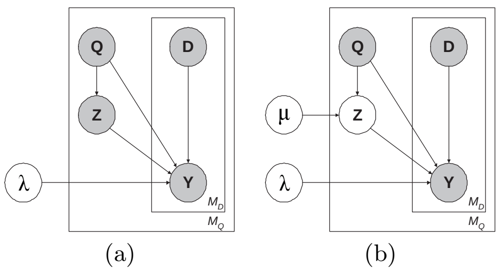


Figure 1: Graphical model representations for (a) the query-class combination where the query classes are manually defined, (b) the probabilistic latent query analysis(pLQA) where the query classes are defined as latent variables. The nodes with known values are shaded, while other nodes are unshaded. The plates stand for replicas of the subgraphs where the number of replicas is on the corner.

following assumptions in our models: (1) the entire query space can be described by a finite number of query classes, where queries from each class share the same combination function; (2) the query description can be used to indicate which class a query belongs to. Under the first assumption, the basic probabilistic retrieval model expressed in Eqn(1) can be naturally extended to a finite mixture of conditional probabilistic models. Formally, we can introduce a multinomial latent query class variable z to indicate which mixture the combination function is drawn from. Based on the second assumption, the choice of z is solely depending on the query Q. Putting all these together, we have the joint probability of relevance y and latent variable z as,

$$P(y_+, z|Q, D; \mu, \lambda) \quad = \quad P(z|Q; \mu)P(y_+|Q, D, z; \lambda), \quad (2)$$

where μ is the parameter for multinomial distributions, λ is the combination parameter for query classes. The mixture component $P(y_+|Q, D, z; \lambda)$ corresponds to a single logistic regression model and the mixing proportion $P(z|Q; \mu)$ controls the switches among different classes based on the query-dependent parameters μ_Q. By marginalizing out the hidden variables z, the corresponding mixture model can be written as, (M_z is the number of query classes)

$$P(y_+|Q, D; \mu, \lambda) = \sum_{z=1}^{M_z} P(z|Q; \mu) \cdot \sigma\left(\sum_{i=1}^{N} \lambda_{zi} f_i(D, Q)\right). \quad (3)$$

In the following discussions, we refer the model presented in Eqn(3) to as the *basic pLQA*(**BpLQA**) model where each latent query class represents a group of similar queries sharing the same combination weights. Note that when the number of latent variables is reduced to 1, BpLQA degrades to the case where retrieval source combination is not relevant to queries, i.e., a query-independent combination approach.

Figure 1(ab) compare the probabilistic graphical model representations of BpLQA and the query-class combination methods, where one of the their major differences is the semantic of the mixing proportions $P(z|Q, \mu)$. In the query-class method, query classes have to be derived from manually defined generation rules before the learning process. However, query classes in the BpLQA model are expressed as latent variables and thus can be estimated directly from training data together with the combination weight estimation. Moreover, they also differ in the way how they associate queries with query classes. By analogy, the query-class method can be viewed as a hard version of "query categorization" which associates each query with one single query class, and meanwhile BpLQA can be viewed as a soft version of "query categorization" which leads to a probabilistic membership assignment of queries to latent query classes.

Because of the representation differences, BpLQA can exploit the following advantages over the query-class combination method: (1) it can automatically discover the query classes from training data rather than by manual definition, (2) it offers probabilistic semantics for the latent query classes and thus allows mixing multiple query types for a single query, (3) it can discover the number of query types in a principled way, (4) it can address the insufficient data problem caused by ill-defined query classes that have few positive examples in the training set and (5) it unifies the combination weight optimization and query class categorization into a single learning framework.

The parameters in BpLQA can be estimated by maximizing its incomplete data log-likelihood. A typical approach to achieve this is to use the Expectation-Maximization (EM) algorithm [3], which can obtain a local optimum of log-likelihood by iterating an E-step and an M-step until convergence. The E-step of the BpLQA model can be derived by computing the expectation of z,

$$h_{tj}(z) = P(z|Q_t, D_j) = \frac{\mu_{zt}\sigma(y_{tj}\sum_i \lambda_{zi} f_i(Q_t, D_j))}{\sum_{z=1}^{M_z} \mu_{zt}\sigma(y_{tj}\sum_i \lambda_{zi} f_i(Q_t, D_j))},$$

where $\mu_{zt} = P(z|Q_t; \mu)$ is the probability of choosing hidden query classes z given query q_m and λ_{zi} is the weights for $f_i(\cdot)$ under the class z. By optimizing the auxiliary Q-function, we can derive the following M-step update rules,

$$\lambda_{zi} \quad = \quad \arg\max_{\lambda_k} \sum_{jt} h_{tj}(z) \log \left[\sigma\left(\sum_{i=1}^{N} \lambda_{zi} f_i(Q_t, D_j)\right)\right],$$

$$\mu_{zt} \quad = \quad \frac{1}{M_D} \sum_{j=1}^{M_D} P(z|Q_t, D_j).$$

When the log-likelihood converges to a local optimum, the estimated parameters can be plugged back into the BpLQA model to describe the underlying query space structure and show how the training queries are organized.

The choice of mixture number K might depend on the amount and the distribution sparseness of the training data. As the number of latent variables grows, the family of decision functions represented in BpLQA will become less restricted and thus lead to lower bias in the estimated models, but meanwhile it would suffer from a higher variance due to an increased model complexity. Based on the bias-variance trade-off, we can determine the number of query classes that are sufficient to capture the variations in the query space with the given amount of training data. To be more concrete, the number of query classes can be obtained by maximizing the sum of the log-likelihood and some model selection criteria such as Akaike Information Criteria(AIC), Bayesian Information Criteria(BIC) and so forth. In our work, we choose BIC [14] as the selection criterion. It quantifies the relative goodness-of-fit of statistical models by maximizing the formula of $BIC = 2l(\mu, \lambda) - k\log(n)$, where k is the number of parameters to optimize and n is the number of training data.

3.2 Adaptive pLQA

Discovering the underlying structure of query space by itself is not sufficient to handle the retrieval source combination, because a practical combination model should be able to predict combination parameters for unseen queries outside the training collection. Unfortunately, BpLQA cannot easily generalize the multinomial parameters μ to any of these unseen queries, because each parameter $\mu_{\cdot t}$ in BpLQA specifically corresponds to the t^{th} training query. Since it is impossible to enumerate all possible queries in the training set, we need to come up with a solution to predict the mixing proportions $P(z|Q_t;\mu)$ of any unseen queries that do not belong to the training collection. To generalize the parameters to new documents, Hofmann [6] suggested a "fold-in" process for the latent class model by re-learning all training documents with the new document to generate an updated parameter estimation. However, the "fold-in" process by plugging in new queries and re-estimating the entire model is not reasonable in our task, because it requires a long time to process and more importantly, we do not have any relevance judgment for new queries to learn from.

To address this problem, we propose an adaptive approach aiming at parameterizing the mixing proportion $P(z|Q_t;\mu)$ using a specific set of features directly extracted from query topics, or called query features, that are able to capture important characteristics of users' information need. Formally, we can represent each query as a bag of query features $\{q_1, ...q_L\}$. The mixing proportions $P(z_k|Q;\mu)$ can then be modeled using a soft-max function $\frac{1}{Z}\exp(\sum_l \mu_{zl}q_l)$, where $Z = \sum_z \exp(\sum_l \mu_{zl}q_l)$ is the normalization factor that scales the exponential function to be a probability distribution. By substituting the mixing proportion back into Eqn(3), previous BpLQA model can be rewritten as,

$$P(y_+|Q,D) = \frac{1}{Z}\sum_z \exp(\sum_l \mu_{zl}q_l)\sigma\left(\sum_{i=1}^{N}\lambda_{zi}f_i(Q,D)\right). \quad (4)$$

Note that, since μ_{zl} is associated with each query feature instead of each training query, this modification allows the estimated μ_{zl} to be applied in any unseen queries as long as they can be formulated as vectors of query features. In the following discussions, we refer the model expressed in Eqn(4) to as the *adaptive pLQA*(**ApLQA**) model.

For this model, the EM algorithm can be derived similarly except the mixing proportions need to be substituted with Eqn(3.2). Therefore, the E-step computes the posterior probability of latent variable z given Q_t and D_j as follows,

$$h_{tj}(z) = \frac{\exp(\sum_l \mu_{zl}q_{tl})\sigma(\sum_i y_{tj}\lambda_{zi}f_i(Q_t,D_j))}{\sum_z \exp(\sum_l \mu_{zl}q_{tl})\sigma(\sum_i y_{tj}\lambda_{zi}f_i(Q_t,D_j))}.$$

In the M-step, we have the same update rule for updating λ_{zi}. For updating μ, we have

$$\mu_{zl} = \arg\max_{\mu_{zl}} \sum_{zt}\left(\sum_j h_{tj}(z)\right)\log\left[\frac{1}{Z_t}\exp(\sum_{l=1}^{L}\mu_{zl}q_{tl})\right],$$

where Z_t is the normalization factor for query Q_t, i.e., $Z_t = \sum_z \exp(\sum_l \mu_{zl}q_{tl})$. The M-step can be optimized by any gradient descent method. Note that, this step is actually fitting a multi-class logistic regression model with query features as inputs. In more detail, it is looking for the most similar logistic regression model w.r.t. the posterior probabilities of z for query Q_t.

The only remaining issue for defining the ApLQA model is to design a set of predictive query features. There are two useful principles to guide the design of suitable query features: 1) they should be able to be automatically generated from query descriptions or the statistics of ranking features, and 2) they should be predictive to estimate which latent classes the query belong to. For example, we can consider the presence/absence of specific person names in query topics, and the mean retrieval scores of each retrieval source as query features. Note that, in contrast to creating query classes that must be mutually exclusive, defining query features is much more flexible, eliminating the need to partition the query space into non-overlapping regions. Moreover, the number of query features can be much larger than the number of query classes with the same amount of training data.

3.3 Kernel pLQA

By introducing explicit query features into the combination function, ApLQA can handle unseen queries that do not appear in the training data. However, the assumption of linear query feature combination in ApLQA might not be the best choice in general because the number of query features is often limited. Also, there exists some useful query information that cannot be described by explicit query feature representation. For example, the edit distance between two queries is a helpful hint for combination but it cannot be easily represented as a explicit query feature. Therefore, we develop an extension of the ApLQA model called the *kernel pLQA*(**KpLQA**) model that lends itself to the use of implicit feature space via Mercer kernels based on the representer theorem [9]. This extension is also motivated by a significant body of recent work that has demonstrated kernel methods are effective in a variety of applications.

In more detail, the kernel representation allows simple learning algorithms to construct a complex decision boundary by projecting the original input space to a high dimensional feature space, even infinitely dimensional in some cases. This seemingly computationally intensive task can be easily achieved through a positive definite reproducing kernel K and the well-known "kernel trick". To begin, let us rewrite the sum term $\sum_l \mu_{zl}q_l$ in Eqn(4) to be an arbitrary function $f_z(Q)$ with respect to the query Q,

$$P(y_+|Q,D) \propto \sum_z \exp f_z(Q)\cdot\sigma\left(\sum_{i=1}^{N}\lambda_{zi}f_i(Q,D)\right). \quad (5)$$

Since the loss function above only depends on the value of f at the data points $\{f_z(Q)\}$, according to the representer theorem, the minimizer $f(x)$ admits a representation of the form $f_z(Q) = \sum_{k=1}^{M_D}\alpha_{zk}K(Q,Q_k)$, where $\{Q_k\}$ is the set of training queries, α_{zk} is the kernel fusion parameter for z and Q_k, and $K(\cdot,\cdot)$ is a Mercer kernel on the query space which has to be positive definite. With this kernel representation, we can derive the corresponding log-likelihood function by substituting original mixing proportion term $P(z|Q_t)$ to be,

$$P(z|Q_t) = \frac{1}{Z}\exp(\sum_{k=1}^{M_D}\alpha_{zk}K(Q_t,Q_k))$$

ApLQA is a special case of KpLQA if each element of K is chosen to be the inner product between the query features of two queries. However, the flexibility of kernel selection

Data Set	**t02s**	**t03s**	**t04s**	**t05s**	-
Query Num	25	25	24	24	-
Doc Num	24263	75850	48818	77979	-
Data Set	-	**t03d**	**t04d**	**t05d**	**t04dx**
Query Num	-	25	24	24	88
Doc Num	-	47531	124097	74532	124097

Table 1: Labels of the video collections and statistics. $t**d$ indicate development sets and $t**s$ indicate search sets where the embedded number is the year. $t04dx$ is the external development set.

has offered more powers to the KpLQA model. For example, the kernel function can have different forms such as the polynomial kernel $K(u, v) = (u \cdot v + 1)^p$ and the Radial Basis Function (RBF) kernel $K(u, v) = \exp(-\gamma \|u - v\|^2)$. The latter one has the ability to project the query features into an infinite dimensional feature spaces. Moreover, we can transform the distance metric between queries (e.g., edit distance between queries) into the implicit feature space in form of a Mercer kernel, instead of designing explicit features for each query. The optimization of KpLQA is similar as that of ApLQA except for changing the mixing proportion term in the E-step and M-step to the kernelized one.

4. EXPERIMENTS

In this section, we present the experimental results on two retrieval applications based on the retrieval source combination, i.e., multimedia retrieval and meta-search.

4.1 Application I: Multimedia Retrieval

Our experiments are designed based on the guidelines of the manual retrieval task in the TREC video retrieval evaluation(TRECVID), which requires an automatic video retrieval system to search relevant documents without any human feedback. In this task, the retrieval units are video shots defined by a common shot boundary reference. The proposed combination algorithms are evaluated using the queries and the video collections officially provided by TREC '02-'05 [16]. For TREC'03-'05, each of these video collections is split into a development set and a search set chronologically by source. The development sets are used as the training pool to develop automatic multimedia retrieval algorithms and the search sets mainly serve as the testbeds for evaluating the performances of retrieval systems. For each query topic, the relevance judgment on search sets was provided officially by NIST. The relevance judgment on development sets was collaboratively collected by several human annotators using the Informedia client [5]. Moreover, we manually designed 40 additional queries and collected the ground truth on the TREC'04 development set in order to demonstrate ApLQA is able to learn from arbitrary queries that do not belong to the testing set[1]. We couple them with the TREC'04-'05 query topics to construct an external training corpus with 88 queries. Table 1 lists the labels of video collections and their query/document statistics.

As building blocks for multimedia retrieval, we generated a number of ranking features on each video document, including 14 high-level semantic features learned from development data (face, anchor, commercial, studio, graphics, weather, sports, outdoor, person, crowd, road, car, building, motion), and 5 uni-modal retrieval experts (text retrieval, face recognition, image-based retrieval based on color, texture and edge histograms). The detailed descriptions on the feature generation can be found in [5]. In an attempt to incorporate the ranking information in the learning process, we weighted the positive data stronger to balance the positive/negative data distribution and meanwhile shifted the median value of each feature to zero [19]. To initialize the EM algorithm in pLQA, we set the mixing parameters μ_z to some random numbers and estimated the combination parameters λ_z from M_z individual queries, which are automatically selected using a simple heuristic similar to the maximal margin relevance.

In order to implement the ApLQA and KpLQA model, we also designed the following binary query features, i.e., if the query topic contains 1) specific person names, 2) specific object names, 3) more than two noun phrases, 4) words related to people/crowd, 5) words related to sports, 6) words related to vehicle, 7) words related to motion, 8) similar image examples w.r.t. color or texture, 9) image examples with faces and finally a feature indicates 10) if the text retrieval module finds more than 100 documents. All these query features can be automatically detected from the query description through some manually defined rules plus natural language processing and image processing techniques. For example, the first three query features can be obtained by the methods of named entity extraction, NP tagging and shallow parsing, of which the details can be found in our previous work [20][2]. The features of sports, vehicle and people can be detected using a manually defined vocabulary with a limited set of words. The eighth and ninth query features can be automatically generated by existing image processing and face detection techniques. Finally the last query feature can be simply obtained from text retrieval statistics. Note that our current implementation requires some query features to be generated from manually defined rules. It is an interesting direction to explore fully automatic approaches to extract query features but we leave it as our future work.

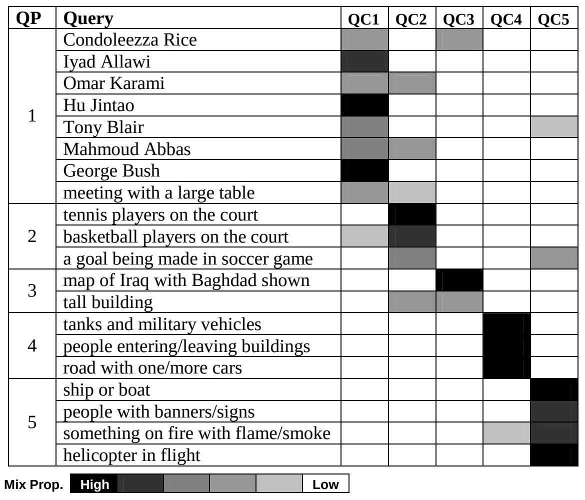
4.1.1 Illustration of latent query classes

To illustrate the ability of pLQA to discover meaningful latent query classes, Figure 2 shows five latent query classes that are automatically learned from the BpLQA model on the TREC'05 development data. In this figure, all of the TREC'05 queries are listed except those without any training data in the development set. The blocks on the right show the scale of mixing proportions $p(z|Q)$ of each query Q w.r.t. each query class z. These queries are organized into five groups based on the class IDs of their maximal mixing proportion in all query classes, i.e., $\arg\max_z p(z|Q)$.

It is interesting to examine whether the query grouping suggested by BpLQA are sensible. Among these five groups, the first one mainly contain all the "named person" queries in the training data and one query related to person appearance, "meeting with a large table". This group of queries usually has a high retrieval performance when using the text features and prefers the existence of person faces, while content-based image retrieval is not effective for them. The second group consists of three queries related to sport events

[1]These queries bring some (but insignificant) performance improvement in our retrieval experiments, as can be verified by comparing performance between BpLQA and ApLQA.

[2]This work [20] also provides evidence that above query features can be predicted with an accuracy above 93%, which is sufficient to support the following retrieval process.



QP	Query	QC1	QC2	QC3	QC4	QC5
1	Condoleezza Rice	░		░		
	Iyad Allawi	▓				
	Omar Karami	░	░			
	Hu Jintao	█				
	Tony Blair	▒				⋅
	Mahmoud Abbas	▒	░			
	George Bush	█				
	meeting with a large table	░	⋅			
2	tennis players on the court		█			
	basketball players on the court	⋅	▓			
	a goal being made in soccer game		▒			░
3	map of Iraq with Baghdad shown			█		
	tall building		░	░		
4	tanks and military vehicles				█	
	people entering/leaving buildings				█	
	road with one/more cars				█	
5	ship or boat					█
	people with banners/signs					▓
	something on fire with flame/smoke				⋅	▓
	helicopter in flight					█

Mix Prop.	High █	▓	▒	░	⋅	Low

Figure 2: We organize TREC'05 queries into five groups (QP) based on which query class (QC) has the highest mixing proportions. The left two columns show group IDs and queries. The other columns indicate the query's mixing proportions w.r.t. five latent classes discovered by BpLQA, where darker blocks mean higher value.

including "basketball", "soccer" and "tennis". They often rely on both text retrieval and image retrieval results, because these sport scenes in the news broadcast usually share common transcript words and image background. The last three groups all contain queries related to object finding, however, they have many distinctions in the combination strategies. In the third group, the queries tend to search for objects that have similar visual appearances without any apparent motions. The case of the fourth group is quite different. They are mainly looking for the objects in the outdoors scene such as "road" and "military vehicle". Finally, the fifth group seems to be a general group that contains all remaining queries. The queries in this group search for objects without visual similarities and thus place a high weight on the text retrieval since text retrieval is usually the most reliable retrieval component in general.

As can be found, most of the discovered latent query classes are consistent with the observations made by many previous studies [20, 1], such as the classes of named person, objects and sports. Meanwhile, BpLQA is also able to suggest some query classes that have not been suggested before such as the class of "outdoor objects". This analysis shows that BpLQA can achieve a reasonable query classification through a completely data-driven statistical learning approach instead of a manual handcrafting procedure.

Finally, it is also helpful to analyze the query mixing patterns learned from BpLQA. For example, the query "Omar Karami" can be described by a mixture of the first and the second query classes. It is not obvious why the second query class is related at first sight. But after a further analysis, we found that Karami always showed up with a similar background when he was meeting foreigners and hence visual appearance turned out to be a helpful clue to find him. Such a mixture treatment provides more flexibilities in retrieval modeling and offers deeper understandings on the queries.

4.1.2 *Retrieval Results*

Next we present the multimedia retrieval results on all

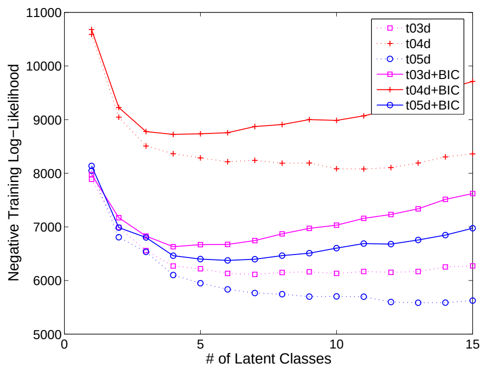


Figure 3: The negative training log-likelihood of BpLQA against the number of query classes. The dash lines indicate the original log-likelihood and the solid lines indicate the regularized log-likelihood with BIC.

the search sets using the proposed models and parameters learned from the development set. Since BpLQA cannot generalize its parameters to unseen queries, we use the development set on the same year as the search set to estimate its parameters. For example, the parameters estimated on the set $t03d$ is reused in the search set $t03s$. For ApLQA and KpLQA, there is no such a constraint and therefore the parameters are estimated with a larger training set $t04dx$.

As discussed before, we can obtain the number of query classes by optimizing the regularized log-likelihood with an additional BIC term. Figure 3 plots the curves of the negative log-likelihood and their regularized counterparts on three development sets (i.e., t03d, t04d and t05d) against the number of query classes. It can be observed that when the number of query classes grows, the learning curves become consistently lower and lower until they asymptotically reach saturated levels. Occasionally the curves even slightly raise at the end, indicating the training data seem to be overfitted in the case of a large number of query classes. Based on the statistical model selection principle, the number of query classes for each training set can be determined by seeking the lowest point on the regularized log-likelihood. Given the ;earning curves, we can find that the optimal numbers of latent classes turn out to be 4, 4 and 6 for the collections of t03d, t04d, t05d respectively.

Table 2(a) lists a detailed comparison between BpLQA with the estimated number of query classes and several baseline methods including text retrieval (Text), query independent (QInd) and query-class combination (QClass) methods with five classes defined in [20]. QClass has shown to be one of the best overall retrieval systems in the past TRECVID evaluations. The parameters in all baseline methods were learned using the same training sets as BpLQA. All the results results are reported in terms of the mean average precision(MAP) up to 1000 documents, precision at top 30, 100 documents and recall at top 1000 documents. To analyze the results in more detail, we also grouped the queries in each collection and reported their MAPs in five different categories, i.e., named person, special object, general object, sports and general queries. As can be observed from the results, QClass is usually superior to both QInd and Text. On average, it brings a roughly 3% absolute improvement (or 30% relative improvement) over the text retrieval. As compared to QClass, BpLQA achieves another 2% boost in

(a)

Data	Method	MAP	P30	P100	R1k	Person	SObj	GObj	Sport	Other
t03s	Text	0.146(+0%)	0.171	0.118	0.477	0.371	0.230	0.068	0.031	0.007
	QInd	0.150(+2%)	0.212	0.136	0.520	0.251	0.293	0.095	0.101	0.012
	QClass*	0.173(+19%)	0.207	0.129	0.531	0.371	0.307	0.091	0.088	0.009
	BpLQA**	**0.205(+40%)**	**0.241**	**0.145**	**0.565**	0.450	0.361	0.102	0.100	0.011
t04s	Text	0.078(+0%)	0.178	0.107	0.361	0.188	0.012	0.033	0.046	0.044
	QInd	0.079(+0%)	0.177	0.116	0.375	0.144	0.063	0.034	0.108	0.051
	QClass	0.087(+11%)	0.186	0.115	**0.380**	0.194	0.080	0.030	0.090	0.044
	BpLQA	**0.104(+33%)**	**0.201**	**0.124**	0.379	0.248	0.084	0.035	0.118	0.047
t05s	Text	0.073(+0%)	0.207	0.175	0.339	0.141	0.015	0.097	0.075	0.016
	QInd*	0.105(+44%)	0.268	0.205	0.387	0.164	0.029	0.090	0.271	0.017
	QClass*	0.108(+47%)	0.261	0.200	**0.396**	0.171	0.032	0.061	0.309	0.018
	BpLQA**	**0.128(+75%)**	**0.307**	**0.212**	0.388	0.212	0.039	0.100	0.316	0.018

(b)

Data	Method	MAP	P30	P100	R1k	Person	SObj	GObj	Sport	Other
t02s	Text	0.098(+0%)	0.113	0.076	0.421	0.124	0.183	0.079	0.023	0.011
	QClass-x	0.132(+34%)	0.131	**0.086**	0.425	0.316	0.207	0.075	0.004	0.013
	ApLQA	0.139(+41%)	0.136	0.084	0.454	0.364	0.194	0.078	0.014	0.017
	KpLQA-R	0.140(+42%)	**0.139**	0.084	0.450	0.365	0.201	0.075	0.014	0.017
	KpLQA-P	**0.144(+46%)**	**0.139**	0.083	**0.462**	0.388	0.206	0.070	0.023	0.019
t03s	Text	0.146(+0%)	0.171	0.118	0.477	0.371	0.230	0.068	0.031	0.007
	QClass-x*	0.200(+37%)	0.236	0.137	0.542	0.407	0.336	0.088	0.106	0.015
	ApLQA*	**0.210(+44%)**	**0.249**	**0.144**	0.562	0.463	0.358	0.106	0.103	0.017
	KpLQA-R*	0.207(+42%)	0.248	0.143	**0.565**	0.469	0.340	0.107	0.100	0.015
	KpLQA-P*	0.206(+41%)	**0.249**	**0.144**	0.544	0.467	0.347	0.097	0.107	0.018
t04s	Text	0.078(+0%)	0.178	0.107	0.361	0.188	0.012	0.033	0.046	0.044
	QClass-x	0.094(+20%)	0.199	0.125	0.381	0.194	0.080	0.046	0.108	0.045
	ApLQA*	0.110(+41%)	**0.220**	0.119	**0.381**	0.255	0.053	0.044	0.110	0.051
	KpLQA-R*	**0.111(+42%)**	0.212	0.120	0.379	0.262	0.067	0.040	0.108	0.050
	KpLQA-P	0.109(+40%)	0.213	**0.128**	0.380	0.255	0.030	0.037	0.116	0.055
t05s	Text	0.073(+0%)	0.207	0.175	0.339	0.141	0.015	0.097	0.075	0.016
	QClass-x*	0.116(+58%)	0.292	0.211	**0.402**	0.173	0.031	0.100	0.322	0.017
	ApLQA*	0.129(+77%)	0.294	0.212	0.397	0.209	0.042	0.104	0.328	0.017
	KpLQA-R*	0.129(+77%)	**0.307**	**0.221**	0.396	0.207	0.040	0.108	0.327	0.018
	KpLQA-P*	**0.130(+78%)**	0.290	0.214	0.390	0.211	0.042	0.100	0.331	0.018

Table 2: The comparison of retrieval performance against combination methods. The numbers in brackets show MAP improvement over Text. Bold texts indicate the highest performance in each criterion. * means statistical significance over Text with *p-value* < 0.01 (sign tests) and ** means significance over both Text and QInd. See text for details.

terms of MAP without any manual tuning on the query class definitions. As it results, the differences between BpLQA and Text/QInd become statistically significant in two out of three collections. The major performance growth factor for BpLQA can be traced to a higher precision on the top documents, although the recalls between all these methods do not vary a lot. By comparing MAPs with respect to each query type, we find that BpLQA benefits most from the Person and Special Object type queries, as well as the General Object type in *t05s*. This is because these query types have their information needs clearly defined and thus they are able to be improved by making better use of the training data.

Table 2(b) compares text retrieval and query-class combination (QClass-x) with ApLQA, KpLQA using the RBF kernel with $\gamma = 0.01$ (KpLQA-R), KpLQA using the polynomial kernel with $p = 3$ (KpLQA-P). All the parameters are estimated from the external training set *t04dx*. Each pLQA model is learned with six query classes based on the regularized likelihood estimation. The experimental settings are similar to those presented above except that the results of *t02s* are evaluated in addition. Note that, QClass-x in this table are learned on a larger training set and thus it can outperform its counterparts in the Table 2(a), which shows the importance of sufficient training data. Despite this change, both ApLQA and KpLQA can still outperform QClass-x by a margin of 1-2% (10-20% relatively) w.r.t. MAP. Similarly, the increase in precision is the main advantage brought by the pLQA models. Among these models, KpLQA show some advantages over the ApLQA model in 3 out of 4 collections, although the difference between them is not significant. However, we believe KpLQA has more opportunities to be improved because it is flexible to incorporate the distance-metric-type features and we will explore this choice in the future.

4.2 Application II: Meta-Search

Our next series of experiments are designed based on the application of meta-search that combines multiple search engines on a single text collection. The TREC-8 collection [18] is used as our testbed which contains 50 query topics and around 2GB worth of documents. Each topic consists of both a short topic title and a long topic description. From the submitted outputs provided by all the participants, we extracted the top five manual retrieval systems and top five automatic retrieval systems as inputs of the meta search system. Their system codes are READWARE2, orcl99man, 8manex, CL99XTopt, iit99ma1, pir9Attd, att99atde, ibms99a, ok8amxc, and fub99td respectively. Each system has at best return 1000 documents for each query. The relevance judgment was officially provided by NIST using a pooling method. We adopt the sum normalization scheme [11] which normalizes the sum of scores from each submission runs to be one and shifts minimum to be zero.

Method	MAP	P30	P100	R1k
BOU	0.465(+0%)	0.587	0.414	0.645
CombSUM	0.565(+21%)	0.603	0.420	0.932
CombMNZ	0.536(+15%)	0.603	0.396	0.914
QInd	0.587(+26%)	0.609	0.441	0.914
ApLQA	0.608(+30%)	0.605	0.450	0.942

Table 3: The comparison of meta-search rerformance against different approaches on TREC-8.

We also extracted the following query features for learning the ApLQA model: length of the query title, appearance of named entities in the query and the score ratio between the first ranked document and 50th ranked document for each of the ten systems. Together with a constant term, we generated a total of 13 query features for each query. These features are designed in an attempt to capture information about query difficulties and generalities.

In the following, we examine the performance of various meta-search algorithms that combine all of the retrieval systems. The retrieval performance is averaged over the last 25 queries in terms of MAP, precision at top 30, 100 documents and recall at top 1000 documents. We evaluated the best underlying retrieval system(BOU) in addition to four meta-search strategies, including CombSUM, CombMNZ, query independent combination (QInd) and ApLQA. For those algorithms that require parameter estimation(QInd and ApLQA), we use the first 25 queries as the training data. As shown in Table 3, CombSUM and CombMNZ can improve upon BOU by a margin of around 10% MAP. Between them, the performance of CombMNZ is slightly worse than that of CombSUM. With aid of the training set, QInd that uses flexible wights is superior to CombSUM that fixes equal weights for every retrieval system. Finally, by introducing the query features and allowing the combination weights vary across different queries, ApLQA offers an additional 2% improvement over QInd w.r.t. MAP. This advantage mainly comes from the extra combination flexibilities provided by ApLQA.

5. CONCLUSION

In this paper, we propose a series of new combination approaches called probabilistic latent query analysis (pLQA) to merge multiple retrieval sources, which unifies the combination weight optimization and query class categorization into a discriminative learning framework. Three pLQA models have been discussed which evolve from a basic version(BpLQA) to an adaptive version (ApLQA) that operates on the query feature space and a kernel version (KpLQA) that builds on a Mercer kernel representation. In contrast to the typical query-independent and query-class combination methods, pLQA can automatically discover latent query classes from the training data rather than relying on manually defined query classes. Also, it can associate one query with a mixture of query classes and thus non-identical combination weights. Finally, based on statistical model selection principles, we can obtain the optimal number of query classes by maximizing the regularized likelihood. Our experiments in two large-scale retrieval applications, i.e., multimedia retrieval and meta-search on the TREC collections, demonstrate the superiority of the proposed methods which can achieve significant gains in average precision over the query-independent/query-class combination methods. We expect that future investigation on designing better query features for ApLQA and introducing some distance-metric-type kernels to KpLQA could result in a further improvement on the performance of retrieval source combination.

Acknowledgement

This work was supported in part by the Advanced Research and Development Activity (ARDA) under contract number H98230-04-C-0406 and NBCHC040037, and by the National Science Foundation under Grant No. IIS-0535056.

6. REFERENCES

[1] T. S. Chua, S. Y. Neo, K. Li, G. H. Wang, R. Shi, M. Zhao, H. Xu, S. Gao, and T. L. Nwe. Trecvid 2004 search and feature extraction task by NUS PRIS. In *NIST TRECVID*, 2004.

[2] T. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445, 1996.

[3] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[4] N. Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[5] A. Hauptmann, M.-Y. Chen, M. Christel, C. Huang, W.-H. Lin, T. Ng, N. Papernick, A. Velivelli, J. Yang, R. Yan, H. Yang, and H. D. Wactlar. Confounded expectations: Informedia at trecvid 2004. In *Proc. of TRECVID*, 2004.

[6] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of the 22nd Intl. ACM SIGIR conference*, pages 50–57, 1999.

[7] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *Proc. of the 26th ACM SIGIR*, pages 64–71. ACM Press, 2003.

[8] L. Kennedy, P. Natsev, and S.-F. Chang. Automatic discovery of query class dependent models for multimodal search. In *ACM Multimedia*, Singapore, November 2005.

[9] G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

[10] R. Manmatha, F. Feng, and T. Rath. Using models of score distributions in information retrieval. In *Proc. of the 27th ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[11] M. Montague and J. A. Aslam. Relevance score normalization for metasearch. In *Proceedings of the 10th international ACM CIKM conference*, pages 427–433, New York, NY, USA, 2001.

[12] R. Nallapati. Discriminative models for information retrieval. In *Proc. of the 27th SIGIR conference on Research and development in information retrieval*, pages 64–71, 2004.

[13] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Informaiton Science*, 27, 1977.

[14] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[15] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Text REtrieval Conference*, 1994.

[16] A. Smeaton and P. Over. TRECVID: Benchmarking the effectiveness of information retrieval tasks on digital video. In *Proc. of the Intl. Conf. on Image and Video Retrieval*, 2003.

[17] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proc. of the 18th ACM SIGIR conference on Research and development in information retrieval*, pages 172–179, 1995.

[18] E. M. Voorhees and D. Harman. Overview of the eighth text retrieval conference (trec-8). In *TREC*, 1999.

[19] R. Yan and A. G. Hauptmann. Efficient margin-based rank learning algorithms for information retrieval. In *International Conference on Image and Video Retrieval(CIVR)*, 2006.

[20] R. Yan, J. Yang, and A. G. Hauptmann. Learning query-class dependent weights in automatic video retrieval. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 548–555, 2004.

[21] 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 *Proceedings of the 28th international ACM SIGIR conference*, pages 512–519, New York, NY, USA, 2005. ACM Press.