DYNAMIC MULTIMODAL FUSION IN VIDEO SEARCH

Lexing Xie, Apostol (Paul) Natsev, Jelena Tešić *

IBM T. J. Watson Research Center, Hawthorne, NY

ABSTRACT

We propose effective multimodal fusion strategies for video search. Multimodal search is a widely applicable information-retrieval problem, and fusion strategies are essential to the system in order to utilize all available retrieval experts and to boost the performance. Prior work has focused on hard- and soft- modeling of query classes and learning weights for each class, while the class partition is either manually defined or learned from data but still insensitive to the testing query. We propose a query-dependent fusion strategy that dynamically generates a *class* among the training queries that are closest to the testing query, based on light-weight query features defined on the outcome of semantic analysis on the query text. A set of optimal weights are then learned on the dynamic class, which aims to model both the co-occurring query features and unusual test queries. Used in conjunction with the rest of our multimodal retrieval system, dynamic query classes performs favorably with hard and soft query classes, and the system performance improves upon the best automatic search run of TRECVID05 and TRECVID06 by 34% and 8%, respectively.

1. INTRODUCTION

In this paper we investigate effective fusion strategies for multimodal video search. Multimodal search is a problem widely applicable in real-world information-retrieval tasks, such as web search, meta-search on text documents, searching personal photos, broadcast video collections, and so on. Here *multimodal* refers not only to multiple ranked lists from multiple retrieval experts, but also to multiple query representations such as having both text descriptions and example images in the query. Prior work has shown that effectively combining ranked lists generated from multiple representations and multiple retrieval experts boosts retrieval performance since different retrieval experts tend to have varying performance on (1) different documents in the database (2) different queries. The first property leads to the performance gain for weighted combinations of rank lists [1]; the latter leads to improvements when tuning the combinations strategies with the input query, which emerge as an active research topic in the multimedia community [2, 3, 4, 5].

In most prior work, a set of weights are pre-learned on a training corpus using manually-defined query classes [2, 3] or learned from data from the training queries and their relevance information [4, 5]. A new query would be assigned a set of weights depending on which training query class(es)/clusters it maps to. These approaches focus on the frequent query classes and types, and assume that each query class or cluster can be modeled separately. Note that the co-existence of different query classes or aspects may have non-linear effect on the weight configuration, and the properties of rare queries are usually lost in one or more distant bigger clusters. We propose joint query matching and weight optimization with dynamically formed "query classes/clusters" to overcome these two difficulties, where such clusters are composed of the nearest neighbors of the incoming query in the query feature space. When used in conjunction with the rest of our multimodal retrieval system, dynamic query classes out perform hard and soft query classes, and the system performance improves upon the best automatic search run of TRECVID05 and TRECVID06 by 34% and 8%, respectively.

Section 2 gives an overview of our multimodal video search system. Section 3 discusses several schemes for optimizing combination weights. Section 4 describes the design for each *uni-modal* retrieval engine, optimized separately for performance. Section 5 presents our experimental results, followed by concluding remarks in Section 6.

2. MULTIMODAL VIDEO SEARCH SYSTEM

A component overview of our multimodal video search system is shown in Figure 1. We take multimodal queries with text descriptions and a few visual query examples. Our query analyses include shallow parsing on the text and feature extraction on the visual examples, respectively. These analysis results are then passed on to the text- and visual- based retrieval engines to generate two ranked lists, as described in Sections 4.1 and 4.2. These analysis results are also used to generate two more ranked lists by combining visual concept detector results. The set of relevant concepts (and their weights) for each query is determined by analysing the query text or the visual query examples as described in Section 4.3. The four available ranked lists are then fused using linear combination with weight vector w, i.e., $f(w,q,d) = \sum_{k=1}^{4} w_k f_k(q,d)$. Where query-dependent weights are learned using techniques proposed in Section 3.

3. QUERY-DEPENDENT FUSION METHODS

We consider linear combination for multiple rank-lists. The weight w usually depends on both prior believes on how good each retrieval expert is, and the characteristics of the query q, since not all queries are answered equally well by any expert. Weight-tuning usually involves three sub-tasks: (a) defining and computing query features, as a basis for generalizing query tuning results to an unseen query, (b) comparing queries, so that a new query can be matched to one or more seen queries, and (c) searching for the optimal weights for one or more of the training queries. We will first describe our query features based on the semantic analysis of query text, following by a discussion of three query matching and weight-optimization schemes.

3.1. Query features from semantic annotation

Query text accurately describes the information need in a multimedia query, it also serves as the sufficient criteria for the human judging on whether or not a query is relevant. The visual query examples usually serve to visualize the information need and help instantiate the visual aspect of the search system. But compared to the relevance decision from a human solely judged from query text, they neither introduce new relevant documents nor limit the scope of the set of relevant documents. Most prior work [5, 4] have found that multimedia queries are usually in a few meaningful and distinct dimensions,

*This material is based upon work funded in part by the U. S. Government. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Government.

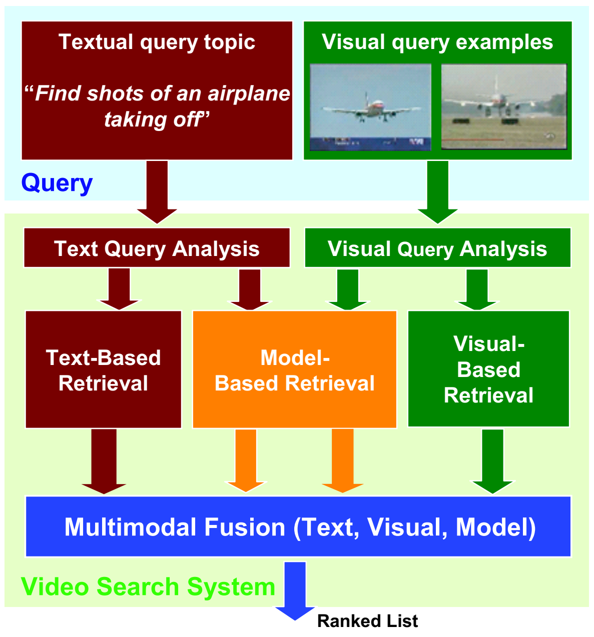


Fig. 1. Multimodal video search system overview.

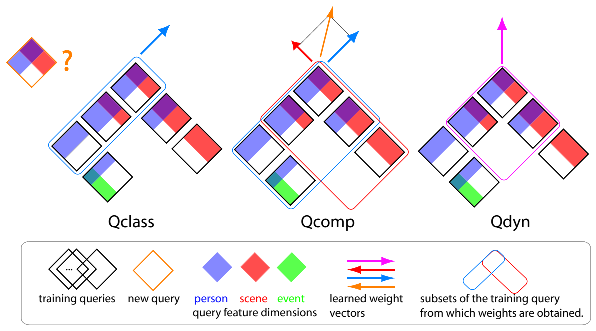


Fig. 2. Illustrative example for the query matching schemes *Qclass*, *Qcomp* and *Qdyn*. See Section 3.2 for explanations.

whether pre-defined or discovered from data. Furthermore, which dimensions each query belong to is often apparent from the query text, such as seeking objects, scene, people, named-people, etc. We analyze the query text in two steps: a semantic analysis is first performed, and then query features are generated from these semantic abstractions.

We use the PIQUANT [6] engine to tag the query text with more than one hundred semantic tags in a broad ontology, designed for question-answering applications on intelligence and news domains. The set of tags cover person, geographic entities, objects, actions, events, etc. For instance, "Hu Jintao, president of the People's Republic of China" would be tagged with "Named-person, President, Geo-political Entity, Nation". Note that in this example, multiple annotations lead to the same visual meaning ("Named-person, President" $\longrightarrow$ person), while some annotations may not have direct visual implications ("Nation"). Hence a *de-noising* step is needed to map these annotations into a few distinct visual categories. We design such a mapping manually from all semantic tags to seven binary feature dimensions, intuitively described as *Sports*, *Named-Person*, *Unamed-Person*, *Vehicle*, *Event*, *Scene*, *Others*. This mapping consists of a few dozen rules based either on commonsense ontological relationship, e.g., a semantic annotation "Person:NAME" leads to *NamedPerson*, or on frequent co-occurence, such as "Road" implies *Vehicle*.

The first of these two steps ensures that we are robust to the variations and noise in natural language, the second step enables us to summarize the semantic abstractions into a few visually meaningful dimensions. Other text-based approaches [5, 4] may take different routes to generate query features, but the outcome for most queries is by and large similar.

3.2. Query matching and weight optimization

Having mapped all queries to a semantic query feature space, we turn to the remaining two sub-tasks: (b) query mapping and (c) weight optimization. The rest of this subsection describes three approaches to completing them, either sequentially or jointly. The first one (*Qclass*) follows the sequence (b)$\longrightarrow$(c), it settles on a few classes/groups and then optimizes weights on each group, as commonly practiced by prior work [4, 2]. The second one (*Qcomp*) extends (b) in *Qclass* to overlapping components, where weights are learned for each component and aggregated for a new query, in similar spirit as [5]. We then propose the third approach (*Qdyn* that generalizes the above by joining (b) and (c) with on-the-fly query mapping and weight optimization. It is also possible to do (c)$\longrightarrow$(b), i.e., learn the best weights for each training query, map new queries to their nearest neighbor and inherit the weights. However this is highly susceptible to noise in each individual query, and we found it of inferior performance in practice.

Qclass: query-class dependent weights. We assign each query into one of seven classes, one for each query feature in Section 3.1. Weights for each class are taken as the set that maximized the average performance metric for all training queries in the class. For non-differentiable performance metrics, this can be done by either exhaustive search on a few dimensions, or heuristic search with restart on a few dozen dimensions.

Qcomp: query-component dependent weights. This extends *Qclass* by allowing overlap in the seven query features. Optimal weights are similarly learned over the set of training queries with this component by maximizing the average performance metric. Weights for a new query is computed as averaging the optimal weights among all of its active components.

Qdyn: dynamically generated query weights. Each new query is mapped to a small set of *neighbors* among the training queries (with inner product of query features serving as the similarity measure), and the weights for the unseen query are obtained by maximizing the average performance on the current set of neighbors. Nearest neighbor mapping performs well in our case, since the query feature space is relatively clean and rather low-dimensional.

Figure 2 illustrates the distinctions among the three schemes, where diamonds denote the queries and filed blocks of different color denote different query features, such as *person*, *event*, *event*. For *Qclass*, we classify queries based on their dominant attribute, e.g., *person* for the test (orange) query and the three training queries to the left. The learned weights (blue arrow) based on these three queries only omits the *scenen* aspect of the testing query hence resulting in sub-optimal weights. *Qcomp* alleviates this problem by modeling the co-occurrences of different query aspects/features, this is done by learning weights for each feature separately (the blue and red arrows) and then use their aggregate for the test query (yellow arrow). This is better than *Qclass* but could still result in in accurate weights since the weights are influenced by dissimilar training features such as *event*. *Qdyn* improves upon both *Qclass* and *Qcomp* by directly modeling the co-occurring aspects of the testing query, it then obtains weights from a set of training queries with the same co-occurring query features.

4. MULTIPLE SEARCH EXPERTS

In this section we summarize each of the text, model and visual uni-modal retrieval component as shown in Figure 1. Details can be found in [7].

4.1. Text-based retrieval

Our text retrieval system was based on the JuruXML semantic search engine [8], a word-frequency based retrieval system with a number of native query expansion functionalities including pseudo-relevance feedback and lexical affinities (i.e., word co-occurrence in close proximity). We build an index of the TRECVID collections from the speech transcript (ASR) or its machine-translated version into English. We time-align the text with the video to generate two index variants: (1) A shot-based baseline index containing words in each sub-shot and five preceding sub-shots aligned with the speaker and phrase boundary that came with the ASR output. (2) A story-based index that takes the words within news story boundaries aligned with speaker/phrase boundaries—this serves to increase recall. In TRECVID06, we use the shot-based index to re-rank shots within stories retrieved using the story-based index, and this generates nearly 30% improvement over using shots alone.

4.2. Visual-based retrieval

The visual based retrieval system is based on the hypothesis that combining two uncorrelated approaches benefits the performance [9], as it combines the high recall of multi-example content based retrieval (MECBR) with high precision of support vector machine (SVM) classifiers. For MECBR, each example was used independently as a CBR query and results were fused using the OR logic (i.e., MAX aggregation of confidence scores). Other parameters (e.g., score normalization) were fixed globally on a feature-dependent but query-independent basis. The realization of SVM discriminative modeling techniques for the search task faces two challenges: very small number of distinct positive examples and no negative examples. We overcome these challenges in two steps: (1) sampling pseudo-negative data points from the video corpus so that they model the test space well, (2) fusing number of primitive SVM predictions trained on the same set of positives and different pseudo-negative sample data using AND logic so that the final SVM model corresponds to the intersection of several positive hyper-spaces derived from the component SVM models. Both approaches were used with five features – global color correlogram, color moments grid, global co-occurrence texture, wavelet texture grid and semantic model vectors. This approach results in a more robust prediction from the visual query examples, as described in [7].

4.3. Model-based retrieval

Model-based retrieval applies the results from off-line concept detection and text analysis to on-line queries by triggering concept models with different weights. Given an arbitrary text- or example-based query, the goal is to identify which visual concepts, if any, are relevant to the query, and to what extent (i.e., what should the weights for each concept be in a weighted fusion scheme). Once the final list of most relevant concept models and weights are determined, we fuse the corresponding concept detection result lists using weighted average score aggregation to generate a final ranked list of shots. For all model-based retrieval purposes we used our detectors for the 39 LSCOM-lite concepts [10].

We employ two query-to-model mapping techniques to generate model-based ranked lists. The first one, code-named *ModelSemantic*, applies concept detection to the visual query examples to generate model vector features. These features are then used as a content-based features for retrieving shots using the same light-weight learning methods used for visual retrieval (Section 4.2). The second one, codenamed *ModelTxt*, determines query-to-concept relevance from the query text alone. This is done using a number of different approaches, including: WordNet-based text-to-model expansion, ontology mapping from semantic annotations in Section 3.1 to a set of concept detectors to trigger via manually defined rules, and pseudo-relevance feedback where the parameters being tuned through the feedback are the weights for the concepts in the lexicon. The ranked lists from all of the above are then averaged to generate one text-based model run.

5. EXPERIMENTAL RESULTS

We evaluate the search fusion approaches on the TRECVID-2005 and 2006 test corpora and query topics. The two collections contain 401 broadcast news video programs from six U.S., Arabic, and Chinese channels, recorded in the fourth quarter of 2004 and 2005, respectively. Each video is 30 minutes to 1 hour each, and the two collections are segmented into 125,249 shots total. Each video comes with a speech transcript obtained through automatic speech recognition, as well as machine translation for the non-English sources. We use Average Precision (AP) at depth 1000 to measure performance on a specific topic, and Mean Average Precision (MAP) to aggregate performance results across multiple topics. Average Precision is the official performance metric adopted by TRECVID [10], which essentially represents the area under the precision-recall curve. Each collection has 24 NIST-distributed queries with pooled ground-truth—we learn the fusion weights on one corpus and apply the weights to the other.

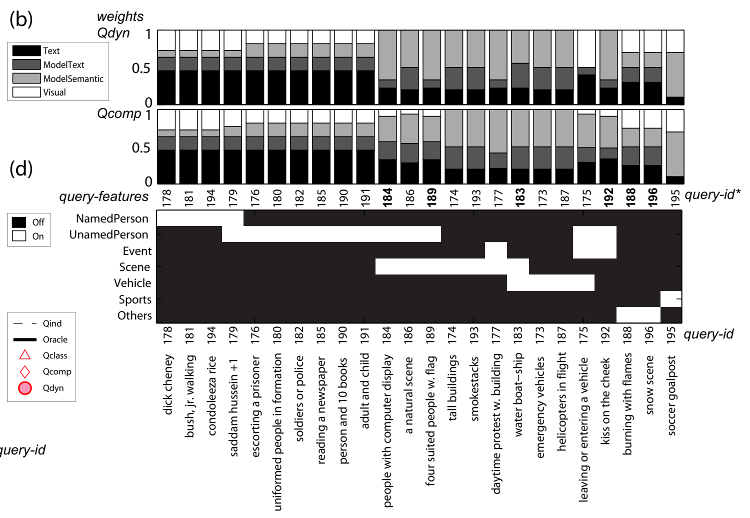
The query-dependent fusion approaches are compared against (1) the query-independent baseline (*Qind*) where a set of optimal weights are obtained from all training queries and then applied to any new query, and (2) an *Oracle* where optimal weights (and hence performance) are obtained for each test query by exhaustive search. The *Oracle* represents the performance ceiling with the current set of uni-modal retrieval experts.

Figure 3(a) shows the MAP statistics for both sets. *Qdyn* and *Qcomp* out performs *Qclass* in general. They also out-perform *Qind* by a margin (results on the three sports queries were unreliable since it over-fitted on the only sports query (195) in TRECVID06). As also shown in Figure 3(c), the MAP, as well as over half of the queries are approaching the *Oracle* performance. The MAPs are already 8% and 34% better over the best MAP among TRECVID06 and '05 automatic retrieval runs of 0.0867 and 0.1190, respectively. Where the improvements on the TRECVID'06 set comes from the novel fusion approaches with the same uni-modal runs, and the improvement from the TRECVID'05 set comes from both the fusion and enhanced uni-modal runs compared to a year ago. Note that given the narrow margin between the baseline and the oracle, our approach shows strong promise.

We can see from Figure 3(d) that the assigned query features concur with our intuitive interpretation of the queries. Furthermore, these queries are clearly partitioned into a few groups represented by one feature (e.g, *NamedPerson*) or the co-occurrence of several feature dimensions (e.g, *UnamedPerson* and *Scene*). Not coincidentally, similar queries in Figure 3(d) have similar weights in Figure 3(b). The automatically learned weights reflect the performance of each retrieval expert on different queries, such as putting more weights on

(a)

dataset	method	Qind	Oracle	Qclass	Qcomp	Qdyn
trec06	all-queries	0.0756	0.1089	0.0897	0.0929	0.0937
	non-sports	0.0695	0.0835	0.0686	0.0720	0.0728
trec05	all-queries	0.1681	0.2184	0.1591	0.1621	0.1604
	non-sports	0.1263	0.1681	0.1291	0.1326	0.1307



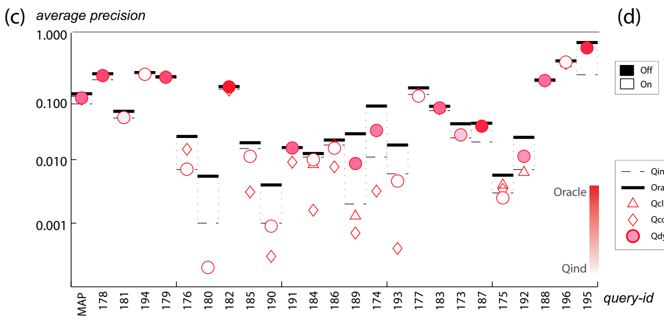


Fig. 3. Result comparison for query-dependent fusion. (a) MAP on TRECVID05 and '06 on all queries and non-*Sports* queries (21 on '05, 23 on '06). (b) Learned weights on TRECVID06 set for *Qdyn* and *Qcomp*, where bold query ids denote improved APs for *Qdyn* over *Qcomp*. (c) AP comparison for all queries in TRECVID06. The tint in the *Qdyn* markers denotes their proximity to the *Oracle* line (percentage in log-scale). (d) Visualization of the query features on the TRECVID06 set. Note that the ordering of all query-ids are according to maximum feature similarity of neighbors in (d).

Text for *NamedPerson* queries, more weights on Models for *Scene* and *Events*, especially when the scene or event is in our visual ontology. For instance, improvements in queries 184, 189 and 192 are accompanied by an increase in weights for *ModelSemantic* and decrease in *Text* and *ModelLexical*, since "computer display", "suits and flags", and "kiss on cheek" are visual concepts rarely mentioned in news anchor speech, and their relation to the concept detectors are better found via visual ontological relationships instead of the generic ontology in WordNet.

These observations demonstrate clear performance gains over the baseline, explain the gains from insights in the data, and show the promise of the proposed query fusion schemes for extending to larger query sets, more complex query feature spaces, and more unimodal retrieval experts.

6. CONCLUSION

We present novel strategies for fusing outcomes from unimodal experts in a multi-source, multi-expert retrieval system. We investigate various query-dependent fusion strategies that dynamically generates a *class* among the training queries that are closest to the testing query, based on light-weight query features defined on the outcome of semantic analysis on the query text. We have evaluated the proposed algorithm on large collections of multi-lingual broadcast news videos. We have observed that dynamic query classes and soft query classes outperform hard query classes, and the system performance improves upon the best automatic search run of TRECVID05 and TRECVID06 significantly. Future work may include extensive analysis of dynamic query classes on a large query collection, and extending the query feature space to accommodate media collections of other genres such as personal photo/blog, sports videos or online video shares.

Acknowledgement

We thank Pablo Duboue and John Prager for their invaluable help on the PIQUANT Semantic Analysis Engine.

7. REFERENCES

[1] E. A. Fox and J. A. Shaw, "Combination of multiple searches," in *Proc. TREC-2*, pp. 243–249, 1994.

[2] R. Yan, J. Yang, and A. G. Hauptmann, "Learning query-class dependent weights in automatic video retrieval," in *ACM Multimedia '04*, pp. 548–555, 2004.

[3] T.-S. Chua, S.-Y. Neo, K.-Y. Li, G. Wang, R. Shi, M. Zhao, and H. Xu, "TRECVID 2004 search and feature extraction task by NUS PRIS," in *NIST TRECVID Workshop*, November 2004.

[4] L. S. Kennedy, A. P. Natsev, and S.-F. Chang, "Automatic discovery of query-class-dependent models for multimodal search," in *ACM Multimedia '05*, pp. 882–891, 2005.

[5] R. Yan and A. G. Hauptmann, "Probabilistic latent query analysis for combining multiple retrieval sources," in *SIGIR '06*, pp. 324–331, 2006.

[6] J. Chu-Carroll, P. A. Duboue, J. M. Prager, and K. Czuba, "Ibm's piquant ii in trec 2005," in *Proc. Fourthteen Text REtrieval Conference Proceedings (TREC 2005)*, 2005.

[7] M. Campbell, S. Ebadollahi, M. Naphade, A. P. Natsev, J. R. Smith, J. Tesic, L. Xie, K. Scheinberg, J. Seidl, and A. Haubold, "IBM research trecvid-2006 video retrieval system," in *NIST TRECVID Workshop*, November 2006.

[8] Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. Maarek, and A. Soffer, "JuruXML—an XML retrieval system," in *INEX '02*, (Schloss Dagstuhl, Germany), Dec. 2002.

[9] A. Natsev, M. R. Naphade, and J. Tešić, "Learning the semantics of multimedia queries and concepts from a small number of examples," in *ACM Multimedia*, (Singapore), 2005.

[10] The National Institute of Standards and Technology (NIST), "TREC video retrieval evaluation," 2001–2006. http://www-nlpir.nist.gov/projects/trecvid/.