Blackthorn: Large-Scale Interactive Multimodal Learning

Jan Zahálka, Stevan Rudinac, Björn Þór Jónsson, Dennis C. Koelma, and Marcel Worring, *Senior Member, IEEE*

***Abstract*—This paper presents Blackthorn, an efficient interactive multimodal learning approach facilitating analysis of multimedia collections of up to 100 million items on a single high-end workstation. Blackthorn features efficient data compression, feature selection, and optimizations to the interactive learning process. The Ratio-64 data representation introduced in this work only costs tens of bytes per item yet preserves most of the visual and textual semantic information with good accuracy. The optimized interactive learning model scores the Ratio-64-compressed data directly, greatly reducing the computational requirements. The experiments compare Blackthorn with two baselines: conventional relevance feedback, and relevance feedback using product quantization to compress the features. The results show that Blackthorn is up to 77.5x faster than the conventional relevance feedback alternative, while outperforming the baseline with respect to the relevance of results: it vastly outperforms the baseline on recall over time and reaches up to 108% of its precision. Compared to the product quantization variant, Blackthorn is just as fast, while producing more relevant results. On the full YFCC100M dataset, Blackthorn performs one complete interaction round in roughly one second whilst maintaining adequate relevance of results, thus opening multimedia collections comprising up to 100 million items to fully interactive learning-based analysis.**

***Index Terms*—Interactive multimodal learning; multimedia analysis; data compression; feature selection; YFCC100M**

I. Introduction

Multimedia collections have become sources of *knowledge* in a large and ever-growing number of scientific and applied domains. A large research effort is being devoted to making the knowledge in large-scale collections more available. The dominant approach revolves around search. Indeed, the search engine is currently the mainstream interface of interaction with multimedia collections. Search-centered retrieval approaches utilize state-of-the-art models, often based on machine learning, to construct a semantic index of the data and offer interactivity by query-response pairs. This is suitable for cases when the user has a clear information need and is able to formulate it as a precise query. However, what if the analyst wants to explore the collection, looking for the question to ask? What if she wants to structure or categorize the data herself, and not according to the structure given by the search engine's index? Such cases require a long process of meaningful *interaction* with the collection in order to *iteratively* build the knowledge according to custom analyst-driven data categorization.

J. Zahálka, S. Rudinac, D. C. Koelma, and M. Worring are with the Informatics Institute, University of Amsterdam, the Netherlands. B. Þ. Jónsson is affiliated with the IT University of Copenhagen, Denmark and Reykjavik University, Iceland. E-mail: j.zahalka@uva.nl; s.rudinac@uva.nl; bjorn@ru.is, d.c.koelma@uva.nl, m.worring@uva.nl.

This paper resulted from an expansion and revision of the 4-page paper titled "Interactive Multimodal Learning on 100 Million Images" by the same authors presented at ACM ICMR '16 [45].

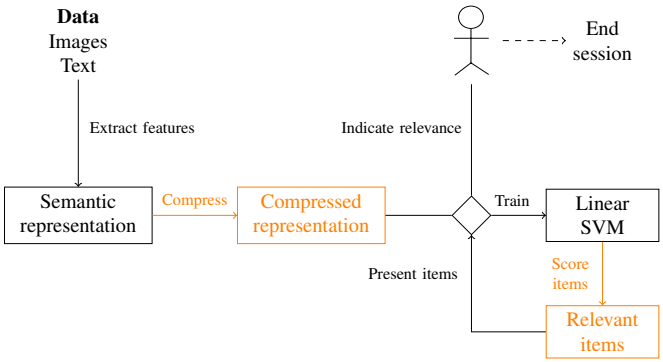

Fig. 1. Interactive learning with Blackthorn. The components of interactive learning innovated by Blackthorn are marked **orange**.

To offer broad analytic capabilities, multimedia systems should support interactive, open-ended tasks where the objective is the analyst's knowledge gain. In her interactive session, the analyst usually alternates between exploration and search, structuring the data in the process. The analyst's knowledge gain is thus captured in a *categorization* of the collection. Notably, for the categorization to capture insight, it must be defined by the analyst herself, and not beforehand by the system [47]. To truly be able to support various interactive solutions for tasks enhancing the analyst's knowledge gain, we set a number of requirements. The first group relates to *performance* and comprises three main requirements:

- *Interactivity*. The user must receive suggested results within seconds at most.
- *Scalability*. The user should be able to interact with even very large-scale (Web-scale) collections.
- *Availability*. The computational resources handling interactions of one user must be modest.

The interactivity requirement enables iterative knowledge gain: by interacting with a responsive interface, the user can build up her knowledge step by step based on the flow of relevant data from the system. Non-responsive interfaces, on the other hand, often result in the user abandoning the analysis altogether. The scalability requirement is especially important in light of the fact that as collections grow to Web scale, obtaining collection-wide human annotations required by supervised approaches becomes infeasible. Semi-supervised, user-driven approaches that scale well do not suffer from the lack of annotations and thus have a potential for large-scale collection analysis. The interactivity and scalability

requirements combined have an implication for computational efficiency: for interactivity to scale properly, the computations must complete in $O(N)$ time in the worst case (where N is the size of the collection). Indeed, given that large-scale collections comprise millions to billions of items, it is difficult to conceive an approach scaling superlinearly with N that completes in interactive time. The availability requirement is set to make interactive knowledge gain approaches accessible on the one hand, and scalable with the number of users on the other. Indeed, it is expensive and wasteful to dedicate the resources of entire computational clusters or distributed systems fully to completing an interactive session of one user. In this work, we consider the availability requirement satisfied when the method operates on a single contemporary high-end workstation (16 cores, 64 GB RAM).

The second group of requirements relates to the *information relevance* for the user and again consists of three requirements:

- *Relevance*. The data items suggested by the machine must be relevant to the user's request.
- *Comprehensibility*. The machine features must have a semantic meaning that the user can understand.
- *Adaptivity*. The user must not be required to follow a data structure or hierarchy defined by the system.

The relevance requirement is, unsurprisingly, a necessity for the knowledge gain: if the user does not receive relevant data, she cannot build up any knowledge, deems the method useless, and abandons the analysis. The comprehensibility requirement relates to the fact that the objective of user knowledge gain cannot be accomplished without the user understanding the data she interacts with. Understanding the features the machine uses for the analysis enables the user to understand the interaction process as a whole, in turn enabling the user to input the invaluable context and her expertise into the process. The adaptivity requirement addresses the need for the user to be able to structure the collection herself. Structuring the collection is an integral part of user knowledge gain [28], [47]. Requiring the user to follow a precomputed structure pigeonholes her into a certain way of thinking, which is undesirable.

Interactive multimodal learning is a good fit for supporting tasks ranging from open-ended exploration to precise search through analytic categorization, which has a machine learning counterpart in classification [47]. Therefore, interactive classification adapting to the analyst's categorization takes the spotlight. Interactive learning sessions, using techniques such as relevance feedback or active learning, elicit training data labels from the user directly and train the model based on them [15]. The user is able to steer the analysis directly through selecting relevant and non-relevant items based on her expertise. The iterative nature aligns with the iterative nature of knowledge build-up. While the interactive learning paradigm is valid, the classic relevance feedback and active learning approach [15], designed for much smaller collections than those we face nowadays, must be revised to fit current needs and meet the six requirements.

Concerning scale, currently the major challenge in the multimedia community is the Yahoo Flickr Creative Commons 100M (YFCC100M) collection, comprising over 12 TB of multimedia data [38]. Its sheer size places a heavy load on computational resources and impedes interactivity. In addition, as collection sizes grow, the discriminativeness of feature vectors decays: more items will have (nearly) identical feature vectors, making them indistinguishable from each other [2], [14], [30]. The body of work on the analysis of the dataset is steadily increasing [18], [20]. YFCC100M has become the central dataset in a number of benchmarks, e.g., the MediaEval Placing Task [22], [9], and the topic of workshops such as MMCommons [5]. Most of the works to date, however, employ entire computational clusters for the analysis, violating the availability requirement.

To the best of our knowledge, no existing approach currently meets all six requirements. How to support interactive solutions for the wide variety of tasks related to knowledge gain in multimedia collections satisfying the six requirements is the research question addressed in this work. Therefore, we develop Blackthorn, an efficient interactive multimodal learning approach for large-scale collections originally introduced in our previous work [45]. Blackthorn, conceptually depicted in Figure 1, brings the following contributions enabling it to fulfill all six requirements outlined above:

1) A compression method reducing the size of semantic representations by up to order of hundreds while preserving most of the semantic information and reducing the number of parameters to minimum.
2) Incorporation of feature selection methods improving the information preservation capabilities of the compressed representation.
3) Revision of the interactive learning process for the large-scale case: building up on the compressed representation and incorporationg good practices, the computational complexity is reduced such that large-scale analysis ($\sim$100 million multimedia items) completes in seconds.

The rest of this work is organized as follows. Section II summarizes related work. Section III presents the data compression method. Section IV describes the adaptation and optimizations to the interactive learning process. Section V outlines the protocol of the experimental evaluation of our method, while Section VI discusses the experimental results. Section VII concludes the paper.

II. Related Work

In response to the rapid growth of digital collections, a number of approaches facilitating increasingly efficient search and exploration have been proposed. Informedia is an early example of a system enabling retrieval, summarization and organisation of video collections [41]. In the years that followed, interactive learning (e.g., relevance feedback, learning new concepts on the fly) has been proven invaluable for improving multimedia information retrieval [15]. The research efforts of the community on interactive search and browsing were concentrated around benchmark initiatives such as VideOlympics [34] or the Video Browser Showdown [33]. However, the collections considered in these benchmarks are much smaller than the one considered in this work.

Most of the state-of-the-art approaches employ entire computational clusters to facilitate interactive or near-interactive

analysis of the data. This is true not only for indexing-based approaches such as the content-based indexing of YFCC100M [18], but also for analytics approaches such as active buckets [11]. The algorithmic efficiency component of these approaches boils down to reducing the number of items for which the similarity is computed, reducing the dimensionality of data representation, or simplifying the similarity computation.

Recently, a large body of work has focused on the scalability of image retrieval. Early work demonstrated that exact k-nn search techniques severely suffer from the curse of dimensionality and that their performance degrades rapidly as soon as the dimensionality of the vectors becomes high enough [6]. Around the turn of the millennium, the emphasis therefore turned to approximate methods for trading off speed against accuracy. For example, several recent works from the multimedia and computer vision communities rely on embedding visual feature vectors in binary space to compress the data and reduce the distance computation time [8], [13], [17], [24], [27], [42], [48]. Depending on whether the distances between a query and items in the collection are computed directly in binary space or a real-valued intermediate space, these approaches can be categorised as symmetric or asymmetric. Each category has its own merits: performing distance computation directly in binary space is generally faster, but it sacrifices search accuracy.

LSH is a well-known scalar-based indexing technique [4]; it has been successfully applied to many contexts (e.g., see [29], [36]). The NV-tree is another scalar-based method, which has been shown to significantly outperform LSH [23]; the NV-tree only requires one eighth of of the processing time of LSH, while only storing about 6 bytes per vector.

Product quantization is a family of vector-based quantization schemes, which decompose the high-dimensional space into low-dimensional subspaces that are indexed independently, thus producing very compact code-words [16], [19], [44]. Another approach applies Map-Reduce to index up to 30 billion high-dimensional descriptors in a distributed setting [26]. This is a simple, yet scalable vector-based quantization method which neither requires a conversion to binary space nor a dissection of the high-dimensional descriptors.

While these approaches for scaling up image retrieval were proven effective in k-nearest neighbour search, their applicability in analytic tasks remains limited for at least two reasons. First, state-of-the-art k-NN search approaches such as binary hashes or product quantization are not suited for classification, which is an essential component of most analytic platforms. Moreover, in many analytic tasks search and exploration are alternately performed, which requires updating, summarizing and re-partitioning of the collection based on user interactions [47]. Such operations require preservation of the original vectors or significant portion of information contained in them. The existing approaches do not focus on analytics, but rather establishing an efficient search structure over the dataset. Novel approaches are needed for building such representations and deploying them efficiently in analytic tasks.

III. Data Representation

The first step in any analysis of a multimedia collection is constructing the data representation. In the interactive setting in general and multimedia analytics in particular, it is highly desirable that the user can understand the representation. Therefore, this work focuses on semantic representations that assign semantic labels to the data in the visual and text modality, for example visual concepts [32] or LDA topics for text [7]. Additional information channels, such as geo coordinates, time, or metadata are semantic per se and can be represented without compression.

The main problem we face with a multimedia collection with 100 million items is the sheer size of the representation. Assuming 1000 visual concepts, 100 text topics, and 8 B as the size of one floating-point number, representing a dataset of 100 million items requires roughly 880 GB of RAM, which is prohibitive. This size must be drastically reduced for the method to fulfill the scalability and availability requirements. Moreover, the comprehensibility and adaptivity requirements must be borne in mind. In this section, we describe a data compression method that greatly reduces the representation size and satisfies the requirements with only modest information loss.

Fortunately, state-of-the-art semantic label representations of multimedia data are generally effectively sparse, i.e., most of the concept and topic scores are zero or negligible. This allows the utilization of sparse representation practices [43]. Let $\mathbf{x}$ denote a feature vector of length n_f that represents one multimedia item in a particular modality. When using a sparse representation, instead of recording all n_f features, we represent each item X by a tuple (S, I) comprising $S = \{\forall x_i \in \mathbf{x} \mid x_i > 0\}$, the set of non-zero feature scores, and $I = \{\forall i \in [1, n_f] \mid x_i > 0\}$, the feature indices corresponding to the scores in S. In this section, we describe how to further reduce the size of X by feature selection (Section III-A) and how to compress the selected features into a compact, efficient representation (sections III-B and III-C).

We note that sparse representations using semantic labels are not the only ones used by the state of the art in multimedia analysis and related disciplines. This is especially prominent in computer vision: dense representations that represent each item by the output of an intermediate layer in a convolutional neural network are widely used. While we design Blackthorn for sparse representations, in this work we also conduct experiments on Blackthorn using dense visual features for the sake of the completeness of the analysis.

A. Feature Selection

In the general large-scale collection case, the data volume reduction through sparse encoding alone (i.e., encoding $\mathbf{x}$ as (S, I)) is insufficient. Therefore, to provide a truly scalable compressed data representation, we must select only a small number of features per modality to represent each item. Formally, this comprises selecting t_f features according to some feature ordering (denoted $O = \{o_1, o_2, \ldots, o_{n_f}\}$). The process produces a reduced set of feature scores $S' = \{x_{o_1}, x_{o_2}, \ldots, x_{o_{t_f}}\}$ and the corresponding feature IDs $I' =$

$\{o_1, o_2, \ldots, o_{t_f}\}$ for each item in each modality. The adaptivity requirement results in the need for an unsupervised feature selection approach. Given the size of target collections, the approach has to come at minimal computational cost. In this section, we describe the three feature selection approaches of Blackthorn.

The first one is the **top-feature** selection: for each item in each modality, the top t_f features sorted by value are taken to represent the item. This is a very straightforward method aligned with the established practices [43].

The following two feature selection approaches aim to address the value imbalance across different features: some features may tend to score consistently higher than others, but at the same time they might not be the descriptive ones. In the visual domain, this has been identified as an issue associated with semantic concepts [3], [31]. In this work, we combat this value imbalance by considering feature values not only within one item, but also relative to the other values of that feature in the dataset. The **thresholded** selection for each feature f first determines the *significance threshold* for each feature defined as $\theta_f = \mu_f + \sigma_f$, where μ_f and σ_f are the feature mean and standard deviation across all items, respectively. Then, for each item X, we consider only those features x_f that have high values relative to the other items, i.e., those which surpass their respective significance threshold ($x_f \geq \theta_f$). The item is represented by the top t_f features that pass the threshold, sorted by feature value.

The third method is the **TF-IDF** selection, inspired by the eponymous time-tested information retrieval method, which has also been successfully deployed in concept-based video representation [31]. For each feature x_f of item X, the TF-IDF score $\mathrm{tfidf}(x_f)$ is described in Equation 1 ($\mathbf{x}$ is the feature vector, $[\![\cdot]\!]$ is the Iverson bracket, and N is the number of items in the collection C):

$$\mathrm{tfidf}\,(x_f) = x_f \cdot \log\left(1 + \frac{N}{\sum_{\mathbf{x}\in C}[\![x_f > \mu_f + \sigma_f]\!]}\right) \quad (1)$$

The TF portion is equal to x_f, the value of the feature itself. The IDF portion determines the feature's rarity, representing the ratio of the items in the collection where the feature is strongly present (using the same significance threshold as in the thresholded selection): the lower the ratio, the more discriminative the feature is. The features in each item are sorted by $\mathrm{tfidf}(x_f)$ in descending order, and as before, the top t_f features sorted by value are selected to represent the item. The TF-IDF selection takes into account both the feature value itself and the distribution of the feature values for each feature.

B. Compression and Decompression

After the feature selection is completed and the top t_f features per item are selected, the data must be compressed. This section describes the compression method used by Blackthorn, as well as the decompression allowing the reconstruction of the features as dictated by the second requirement for interactive learning representations.

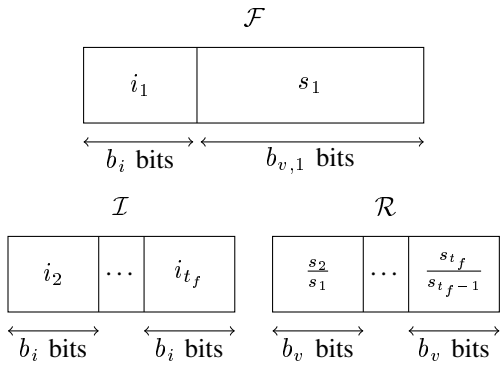

Fig. 2. The ratio representation, encoding the top-valued feature ($\mathcal{F}$), feature IDs ($\mathcal{I}$), and ratios between subsequent features ($\mathcal{R}$).

The memory required for representing the top t_f features in each modality depends on three parameters: the value of t_f, the number of bits required to encode the feature ID (denoted b_i), and the number of bits required to represent the feature value (b_v). The value of b_i in turn depends on n_f, the total number of features in the original feature set: $b_i = \lceil \log_2 n_f \rceil$. The value of b_v depends on the decimal precision p required for the recorded value: $b_v = \lceil p \cdot \log_2 10 \rceil$. Note that this assumes feature values between 0 and 1, which can be easily achieved by normalization.

The straightforward way to sparsely encode features is to directly encode the feature value in one bit field and the feature ID in another. Indeed, this is the paradigm used in the first version of Blackthorn [45]. Whilst this poses no problem for encoding the feature IDs, the selection of p is difficult, which raises an important issue with respect to accuracy and compactness. When p is too low, many features fall below the decimal precision of p and are encoded as zeros, increasing the loss of information and wasting storage space. When p is too high, the information drop is alleviated, but the compactness of the representation suffers. Dynamic p for different items impairs random access to the encoded features for each item. Therefore, a different solution is needed.

To alleviate the decimal precision problem, we introduce the **ratio representation**, schematically depicted in Figure 2. The ratio representation uses three bit fields. The first field, $\mathcal{F}$, encodes the feature identifier and value of the strongest feature directly, as the value of the strongest feature is the starting point of the ratio representation. The decimal precision of the strongest feature's encoding and the corresponding bit cost are henceforth denoted p_1 and $b_{v,1}$, respectively. For each successive feature, the $\mathcal{I}$ bitfield encodes the feature identifiers, and the $\mathcal{R}$ bitfield encodes the *ratio* between the i-th feature score ($2 \leq i \leq n_f$) and the preceding $(i-1)$-th feature score. Thus, the p parameter does not determine the maximal decimal precision of the encoded features anymore, but rather the lowest possible ratio, or the maximal allowed value "drop" between any two features. Note that this makes the p parameter much less constraining, as we are now able to encode feature scores with arbitrary decimal precision, given that the value drop between the features is less than p orders.

The compression operates as follows. Let $b_{v,1}$ denote the

number of bits required to encode the top feature score in the $\mathcal{F}$ bitfield and p_1 the decimal precision pertaining to the encoding of the top feature. Further, let $[\cdot]$ denote the rounding operation, $\vee$ the binary OR operation, and $\ll$ the left bit shift operation. The construction of the $\mathcal{F}, \mathcal{I}$, and $\mathcal{R}$ bit fields is described by Equations 2, 3, and 4, respectively.

$$\mathcal{F} = [10^{p_1} \cdot s_1] \vee (i_1 \ll b_{v,1}) \quad (2)$$

$$\mathcal{I} = \bigvee_{k=2}^{t_f} (i_k \ll (k-2) \cdot b_i) \quad (3)$$

$$\mathcal{R} = \bigvee_{k=2}^{t_f} \left(\left[10^p \cdot \frac{s_k}{s_{k-1}} \right] \ll (k-2) \cdot b_v \right) \quad (4)$$

Most of the original feature vector can be reconstructed by decompressing $\mathcal{F}, \mathcal{I}$, and $\mathcal{R}$, with the features outside the top t_f set to 0. Let $\wedge$ denote the binary AND operation and $\gg$ the right aritmetic bit shift operation. The feature ID decompression for the k-th feature is defined in Equation 5 ($k = 1$) and Equation 6 ($2 \leq k \leq t_f$). The feature score decompression for the k-th feature is defined in Equation 7 ($k = 1$) and Equation 8 ($2 \leq k \leq t_f$).

$$\delta(i_1) = \mathcal{F} \gg b_{v,1} \quad (5)$$

$$\delta(i_k) = (\mathcal{I} \gg ((k-2) \cdot b_i)) \wedge \left(2^{b_i} - 1\right) \quad (6)$$

$$\delta(s_1) = \mathcal{F} \wedge 2^{b_{v,1}} \quad (7)$$

$$\delta(s_k) = \delta(s_1) \prod_{j=0}^{k-2} \frac{(\mathcal{R} \gg j \cdot b_v) \wedge \left(2^{b_v} - 1\right)}{10^p} \quad (8)$$

Note that $\delta(s_k)$ does not allow for equal-time random access to the k-th feature score. This constitutes the computational price for the ratio representation. However, since the compressed representation is designed to be decompressed sequentially to obtain the entire feature vector, this is not an issue: we can simply keep the "running feature score" in one extra variable whilst decompressing. This very minor memory demand is more than outweighed by the advantage of arbitrary decimal precision brought by the ratio representation.

C. The Ratio-64 Representation

Currently, most machines are 64-bit, making it desirable to match the actual size of the $\mathcal{F}, \mathcal{I}$ and $\mathcal{R}$ bitfields as defined by Equations 2, 3 and 4, respectively. Aligning the bitfields with the 64-bit architecture boils down to choosing t_f, p_1, and p accordingly. The selection must be made bearing in mind n_f, the total number of features, and the corresponding b_i.

Encoding $\mathcal{F}$ in one 64-bit integer is more than sufficient, because $b_{v,1} = 64 - b_i$ leaves more than enough bits for high p_1: for example, in the case of feature sets with $n_f \leq 16384$, $b_i = 14$, which makes $b_{v,1} = 50$, allowing the encoding of the top feature score with the decimal precision of 15. Encoding $\mathcal{I}$ is straightforward: each feature ID requires exactly b_i bits, making the total cost of encoding the feature IDs in $\mathcal{I}$ equal to $t_f \cdot b_i$. Encoding $\mathcal{R}$ requires checking the decimal precision required to encode the minimal ratio between any two features in the dataset and setting b_v accordingly. Encoding $\mathcal{R}$ in ι 64-bit integers boils down to encoding $\phi = \lfloor \frac{64}{b_v} \rfloor$ feature score ratios per integer. $\mathcal{I}$ is encoded analogously.

TABLE I
COMPARISON OF THE MEMORY COST OF RATIO-64 AND THE UNCOMPRESSED REPRESENTATION: THE MEMORY COST OF 1 ITEM, THE MEMORY COST OF 100 MILLION ITEMS, AND HOW MANY ITEMS FIT INTO 60 GB.

	1 item	**100M**	**60 GB**
Uncompressed	8.81 kB	880.8 GB	6.8M items
Ratio-64, $\iota = 1$	48 B	4.8 GB	1250M items
Ratio-64, $\iota = 5$	176 B	17.6 GB	340.9M items
Ratio-64, $\iota = 10$	336 B	33.6 GB	178.5M items
Ratio-64, $\iota = 15$	496 B	49.6 GB	120.9M items

The memory-efficient **Ratio-64 representation**, depicted in Figure 3, only requires $2\iota + 1$ 64-bit integers per item and modality. Table I compares the uncompressed representation with the Ratio-64 representation. With $\iota = 1$, the representation is 183.5x smaller than the uncompressed representation and allows fitting 1.25 billion items into 60 GB of memory. The number of preserved features depends on the values of b_i, and b_v, which is intrinsically tied to the data itself. Assuming the space-generous case of $b_i = b_v = 14$ (allowing for $p = 4$ and n_f up to 16,384), ϕ is then equal to 4, enabling the preservation of tens of features per item. The ratio paradigm ensures that arbitrarily low values can be preserved. Note that the features are selected *per item*, not for the entire dataset. Thus, no features get discarded en bloc based on dataset-wide statistics. The value of ι is subject to experimentation in this work. With its design, the Ratio-64 representation aligns with the requirements for large-scale interactive multimodal learning set in the introduction.

IV. INTERACTIVE LEARNING

The compressed representation is a necessary condition for interactive analytics on large scale multimedia collections, but not a sufficient one. To facilitate large-scale interactive multimedia analytics, the interactive learning process itself must be optimized. In this section, we describe the performance optimizations considered in this work.

An interactive multimodal learning session consists of a number of *interaction rounds*, each with 3 steps:

1) An interactive classifier $\mathcal{C}_I$ (per modality) is trained on a set of examples provided by the user.
2) Using $\mathcal{C}_I$, the unlabelled items in the collection are scored.
3) The top r results are returned to the user. Using late modality fusion, this requires to fuse the rankings across modalities to obtain the final top r.

The process requires an *interactive* classifier, i.e., one that can be trained during each interaction round without violating the requirements outlined in the introduction, especially interactivity, relevance, and scalability. As a result, not all classifiers are suitable. For instance, deep nets, which are very much embedded in the state of the art in multimedia analysis, computer vision, and information retrieval, are out

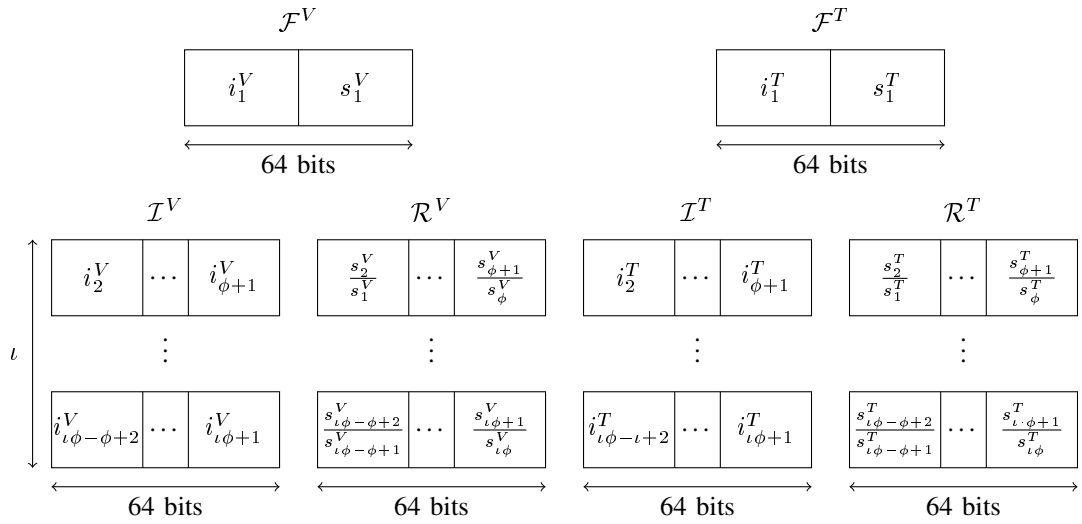

Fig. 3. The Ratio-64 representation encoding the visual (V) and text (T) features. For each modality, $\mathcal{F}$ encodes the first feature value and ID. $\mathcal{I}$ and $\mathcal{R}$ are each a collection of ι 64-bit integers. In the i-th slot, $\mathcal{I}$ encodes the i-th feature ID, while $\mathcal{R}$ encodes the ratio between the i-th and the $i-1$-th feature value.

of the question: they generaly take too long to train on large datasets and require more than a few judgments provided by the user.

A user typically produces tens of examples per interaction round at most. As a consequence, the set of training examples does not scale with the size of the collection (N). We consider two variants of training. The first one is training on the original data, loading the uncompressed features for the training samples on demand. This approach utilizes the full information contained in the features, but random access to the file containing the uncompressed features (as they cannot be kept in RAM) may scale sub-optimally with collection size. The second variant is training on the decompressed Ratio-64 features. This provides faster access, but the training algorithm has access to incomplete feature information due to the information loss incurred by the Ratio-64 representation. The effect of using the decompressed data or original, uncompressed data is subject to experimentation in this work.

The time cost of steps 2 and 3 increases dramatically as the collection grows larger. Compression and efficient implementation do not alleviate this problem on their own. To tackle this problem, Blackthorn considers two optimizations of the interactive learning process: an adaptation of the classifier scoring method exploiting the structure of the compressed dataset, and modality fusion with reduced computational overhead. Both follow the established good practices in the multimedia community.

Adapting the scoring method to suit the compressed Ratio-64 representation means scoring the items directly in the compressed space without expanding each of the vectors into the original vector space. This is straightforward for classifiers that employ a dot product of a weight vector and the feature vector to score an item, such as SVM, perceptron, or linear discriminant analysis. For such classifiers, the dot product computation can be sped up by iterating only over those feature coordinates actually recorded in the Ratio-64 representation of the item, instead of all coordinates of the original feature space. Expanding the compressed vector into the original feature space yields no benefit, since all non-recorded features are 0 and iterating over them has no effect on the value of the dot product.

To demonstrate the speed-up, consider for instance the linear SVM and its scoring function (Equation 9, $\mathbf{w}$ is the SVM weight vector and b the bias):

$$\sigma(I) = \mathbf{w}^T\mathbf{x} + b \tag{9}$$

Assuming equal computational cost of addition and multiplication for the sake of simplicity, this costs $2n_f + 1$ arithmetic operations per item and modality. Given a semantic representation with 1000 visual concepts and 100 text topics, this amounts to 2001 operations for the visual modality and 201 operations for the text modality. Using the Ratio-64 representation, only the $\phi \cdot \iota$ encoded features carry any value. Iterating over those features only yields the scoring function in Ratio-64-compressed space (σ_c) defined in Equation 10:

$$\sigma_c(I) = \sum_{k=1}^{t_f} \left(w_{\delta(i_k)} \cdot \delta(s_k)\right) + b \tag{10}$$

The feature ID decompressions cost 2 mathematical operations each (cf. Equation 6), except the first one, which costs 1 operation (cf. Equation 5). The feature score decompressions cost 4 operations each (cf. Equation 8), except the first one, which costs 1 operation (cf. Equation 7). On top of the decompression, the score computation comprises $t_f + 1$ additions and t_f multiplications. This amounts to $8t_f - 3$ mathematical operations per item and modality. Since $t_f \ll n_f$, σ_c is faster than σ by up to orders of magnitude. For example, in the case of $\iota = 1, \phi = 6$, a visual concept dictionary of 1000 concepts, and text topic dictionary of 100 topics: σ_c performs 37.7x fewer operations in the visual modality and 3.7x fewer operations in the text modality, while producing exactly the same result.

The second aspect requiring attention is late modality fusion, which has been long established to perform better than early fusion [35]. With a straightforward approach, this requires $O(N \log N)$ time, which is higher than the $O(N)$ complexity of scoring. To reduce the total complexity to $O(N)$, we follow the protocol of the top-r list rank aggregation [12] in order to obtain the top r relevant results. For each modality, the top-scoring ν (with $r \leq \nu \ll n$) results in that modality are nominated to the final ranking by iterating over the data once and checking whether each respective item qualifies for the top-ν ($O(N)$ time). The multimodal ranking is obtained by performing rank aggregations on the nominated items ($O(\nu \log \nu)$ time). Because $\nu \ll N$, modality fusion is essentially computed in sublinear time. Optimized modality fusion in combination with efficient scoring ensures that all stages of interactive learning complete in $O(N)$ time.

V. Experimental Setup

In the experimental evaluation of Blackthorn, we aim to answer the following questions:

1) What is the overall speed-up of Blackthorn compared to state-of-the-art methods in the interactive learning setting?
2) How does Blackthorn compare to the state of the art regarding the ability to produce relevant results?
3) Which combination of Blackthorn's parameters yields the best performance?
4) Does Blackthorn work with dense feature representation as well?
5) How does Blackthorn scale compared to state-of-the-art methods in the interactive learning setting?

In order to evaluate the individual optimizations/parameters proposed in this work and their effect on Blackthorn's performance, the experiments are conducted on a number of variants of Blackthorn. The Blackthorn variants based on the optimization used are named `bt_#FEATSEL_#TRAIN`. `#FEATSEL` is a label for the feature selection used (cf. Section III-A), which can have three values: `topft` for the top-feature selection, `thres` for the thresholded selection, and `tfidf` for the TF-IDF selection. `#TRAIN` is a label for the feature variant used for the training of the interactive learning classifier (cf. Section IV), which can have two values: `ucmp` whenever the original, uncompressed features were used for the training and `comp` whenever the compressed features were used.

In the experiments, we compare Blackthorn with two approaches reflecting the state of the art. The first one is the standard relevance feedback approach [15] (henceforth denoted `rf_standard`), which is still being used in the interactive learning scenario to this day. Given that Blackthorn is essentially optimized relevance feedback, sacrificing the accuracy of the feature representation for speed, this allows us to evaluate how favourable Blackthorn's accuracy-speed trade-off is.

The second approach uses product quantization [16], [19], [44] to compress the features. Product quantization is the most applicable state-of-the-art technique in the context of interactive learning, despite the fact that interactive learning cannot make use of its efficient search structure. Following [16], we split the feature vectors in m subvectors, replacing each with the respective cluster ID resulting from a subquantizer partitioning the feature space into k clusters. We opted for $m = 12$ and $k = 1024$. This choice is motivated by the good practices outlined in [16] (large k and small m is better than the other way round). In particular, $m = 12$ ensures `rf_pq` is on par with the efficient configurations on Blackthorn compression-wise, and $k = 1024$ is reasonable regarding both compression quality (cf. [16], Fig. 1) and computational speed.

As the quantization algorithm we use a randomized quantizer that establishes k random samples from the collection as centroids and assigns each item to the closest centroid. In other words, we perform 1-step random-initialized k-means, establishing a Voronoi partition of the feature space. The choice of this simple algorithm is dictated by the dimensionality of quantizing a large-scale dataset, which is both time- and memory-consuming. The resulting representation is then used in a standard relevance feedback setting.

This approach is further denoted `rf_pq`. Comparing Blackthorn with `rf_pq` pits Blackthorn's Ratio-64 representation against a compressed representation widely used in state-of-the-art k-NN search. Note that the features obtained by product quantization violate the comprehensibility requirement. Also note that the experiments are meant to gauge the efficiency of product quantization in the context of interactive learning only, not to evaluate product quantization as a whole.

As the instantiation of the interactive classifier $\mathcal{C}_I$, we use linear SVM [10] for all three approaches. The main reason is that linear SVM still remains the most widely adopted interactive classifier, appearing in a number of recent state-of-the-art works, e.g., [21], [25]. As discussed in Section IV, deep nets, which are currently the mainstream classifier in multimedia analysis and especially computer vision, take too long to train and are thus unsuitable in the interactive setting. All approaches are implemented in C and use the VLFeat implementation of the SVM [39].

Interactive learning in the multimedia analytics context is notoriously difficult to evaluate, due to the difficult benchmarking of insight-driven analysis. Indeed, every user proceeds to different insights through different means, which is difficult to capture in objective benchmarks. Since this work focuses on the learning process in general, and not on a particular interactive system, a user study is not an option, as it is very difficult to isolate the effect of using an improved learning method from the effect of the interface on the user. Therefore, the evaluation of Blackthorn's efficiency is inspired by the Analytic Quality paradigm [46], employing a large number of automated computer agents (or *actors*) simulating user behaviour. These actors simulate the actions of a human user in the classic relevance feedback scenario: each round manually selecting relevant and non-relevant items and submitting the selection to the relevance feedback algorithm, which returns the items deemed most relevant for the next round of the user's consideration.

Since there are no annotated datasets suitable for multimedia

Fig. 4. Example items involved in the first interaction round performed on the YFCC100M dataset by the actor interested in items from Prague. In [45], none of the suggestions were regarded as relevant for the evaluation, despite the semantically similar content.

analytics evaluation, let alone ones on the order of hundreds of millions of items, we resort to a custom evaluation task performed on YFCC100M. The evaluation task is to retrieve items pertaining to a large city in an interactive session based *solely* on visual and text content. This task is inspired by the MediaEval Placing Task [22], [9], one of the main multimedia benchmark challenges revolving around the YFCC100M dataset. However, note that the task is quite difficult. Its only purpose is to compare the general interactive learning approaches with each other. In this work, we do not strive to outperform the state-of-the-art localization approaches built for the purpose of the Placing task.

We define the actors that shall interact with the evaluated algorithms as follows. First, we sort world cities in descending order by the number of associated items in the 2016 test dataset of the Placing task. Each actor then corresponds to one city in the top 50, and its set of relevant items is defined as all items within 1000 kilometers of the city centre, corresponding to the largest radius considered in the Placing task. This large radius has been chosen to counter the problem encountered in the initial iteration of our work [45] and illustrated in Figure 4, where the methods were penalized for returning reasonable, semantically similar suggestions that did not come from exactly the same city, but from a culturally and architecturally close city. For each actor, 50 sessions are conducted. The initial relevance indication in each comprises 100 uniform random samples from the actor's relevant items as positives and 200 uniform random samples from the collection as negatives. In each interaction round, the evaluated algorithm retrains the model and suggests 25 items to the actor, i. e., $r = 25$ for all experiments. Those suggestions that are contained in the actor's relevant item set are added to the set of positives for the next interaction round. The set of negatives for the next round contains 100 uniform random samples from the collection.

Two datasets were used for the evaluation: the Placing 2016 test set and YFCC100M. With 1.5 million items, the Placing test set is rich enough to provide a challenge for the algorithms and reasonably sized to allow for comparison of `rf_standard` with the Blackthorn variants. We experiment on the Placing test set to answer the first four experimental questions. The second dataset, YFCC100M, is used to gauge the scalability of Blackthorn to truly large datasets. Note that YFCC100M is too big to be handled by `rf_standard`. A persistent approach with the data stored on disk is not an option, since the amount of I/O operations would cripple any attempt at interactivity.

To provide a broader, general overview of the algorithms' performance, two different semantic representations were used in each of the two modalities for both datasets. In the visual modality, we extracted the 1000 ILSVRC ImageNet concepts [32] and the 365 Places2 concepts [49], both using a GoogLeNet convolutional neural network [37]. In addition, we also experiment on the dense 1024-dimensional features obtained as the output from the average pooling layer of each respective GoogLeNet. This allows us to showcase Blackthorn on dense features which are commonly used in the computer vision community. However, we consider the dense features auxiliary and focus on the sparse concept features in greater detail. The reason for this is that the average-pooling features violate the comprehensibility requirement posited in the introduction: they are meaningless to the interacting user. In the text modality, we extracted 100 LDA topics directly from the YFCC100M corpus, as well as 100 topics obtained by applying the LDA model constructed on the Wikipedia corpus [1]. The LDA extraction was conducted using the Gensim framework [40].

In the experiments, we vary the ι (cf. Section III-C) and ν (cf. Section IV) parameters: $\iota = \{1, 5, 10, 15\}, \nu = \{100, 500, 1000\}$. Bearing in mind the selected feature representations, we instantiate the Ratio-64 parameters ($b_i, b_v, b_{v,1}, t_f, \phi$, cf. Section III) as follows. We set these parameters equally for both modalities for the sake of simplicity. The highest number of features (n_f) is 1024, corresponding to $b_i = 10$. We set $b_v = 10$ as well, as this is sufficient for encoding all ratios in all the datasets. The $b_{v,1}$ parameter is set to 54, which allows for a very generous encoding precision and uses all 64 bits of the top-valued feature ($\mathcal{F}$) bitfield. Given the b_v, $\phi = 6$ and thus, for each modality, $t_f = 6\iota + 1$.

Further, The ϕ parameter has been set to $\phi = 6$ based on the nature of the data in accordance with the practice outlined

in Section III-C, i. e., checking the decimal precision required to encode $\mathcal{R}$.

To gauge the performance of the evaluated algorithms, we report four evaluation measures reflecting the requirements of an analyst:

- Time per interaction round
- Precision at interaction round
- Recall after 10 interaction rounds
- Recall over time

Time per interaction round is the alpha and omega of interactivity, not merely a measure of computational efficiency or convenience for the user. Indeed, for an algorithm to be interactive, the time per interaction round cannot exceed a couple of seconds, as dictated by the interactivity requirement. The time per interaction round encompasses all three steps defined in Section IV. *Precision at interaction round* reflects the need for the algorithm to produce relevant items in the individual interaction rounds to engage the analyst. Indeed, if the algorithm goes through long dry spells with respect to relevance, the analyst quickly loses confidence in its analytic capability, again increasing the likelihood of the user ending the session prematurely. *Recall after 10 interaction rounds* reflects the cumulative information gain after the early stage of the analysis. However, this does not reflect the time needed to arrive at that particular information gain. *Recall over time* captures both timely analysis and user insight gain. Indeed, the more relevant items the analyst sees and the earlier she sees them, the better. We report the algorithmic recall over time, not taking into account the time required by the user to process the results. This is due to the user processing time being highly volatile and influenced by factors beyond the scope of this work, such as the interface of the system implementing the algorithm.

VI. Results and Discussion

Table II summarizes the precision, recall after 10 interaction rounds, and time per interaction round using sparse visual features, i.e., the visual concept scores. For the sake of brevity, we report up to five results per combination of features: `rf_standard`, the `rf_pq` configuration with the highest precision/recall, the Blackthorn configuration with highest precision, the Blackthorn configuration with highest recall (if it is different from the previous one), and finally the fastest Blackthorn configuration. The results clearly reveal that speed and efficiency are Blackthorn's forte. The fastest configurations of Blackthorn consistently complete one interaction round in under a tenth of a second, which is on par with the slightly faster `rf_pq`, and yields a massive speedup of 61.4–77.5 compared to `rf_standard`'s barely interactive 3–4.5 seconds. Blackthorn's speed-up answers the first experimental question quite convincingly.

Regarding the second experimental question, Table II shows that relevance-wise, Blackthorn is on par or sometimes better than `rf_standard` and much better than `rf_pq` in all cases. This means that the information loss incurred by the Ratio-64 representation is very modest in the worst case and even turned into an information gain in the best case, making the information loss-speed trade-off quite favourable. Out of all four feature set combinations, Blackthorn preserves at least 93% of the precision achieved by `rf_standard` and increases the precision to 108% of `rf_standard`'s in the best case. Regarding recall at 10 interaction rounds, Blackthorn achieves 89–98% of that achieved by `rf_standard`. However, when recall over time is taken into account, Figure 5 clearly showcases that Blackthorn dominates both `rf_standard` and `rf_pq` already from the early interaction rounds. Note that all highest-scoring Blackthorn configurations with respect to precision and recall use either TF-IDF or thresholded feature selection, validating the feature selection approaches presented in Section III-A. Thus, the answer to the second experimental question is that Blackthorn is able to provide relevant results comparable or better than the best baseline, incurring minimal information loss further alleviated by feature selection.

TABLE II
PRECISION, RECALL AT 10 INTERACTION ROUNDS, AND TIME PER INTERACTION ROUND ON THE PLACING TEST DATASET, USING THE SPARSE CONCEPT SCORES AS VISUAL FEATURES.

IMAGENET (CONCEPTS), YFCC100M	**precision**	**recall**	**time**
`rf_standard`	0.176	$6.62 \cdot 10^{-4}$	4.72 s
`rf_pq` ($\nu = 100$)	0.106	$1.61 \cdot 10^{-4}$	0.03 s
`bt_tfidf_ucmp` ($\iota = 1, \nu = 1000$)	0.191	$6.17 \cdot 10^{-4}$	0.09 s
`bt_tfidf_comp` ($\iota = 5, \nu = 1000$)	0.180	$6.19 \cdot 10^{-4}$	0.14 s
`bt_thres_comp` ($\iota = 1, \nu = 100$)	0.154	$4.67 \cdot 10^{-4}$	0.07 s
IMAGENET (CONCEPTS), WIKIPEDIA	**precision**	**recall**	**time**
`rf_standard`	0.214	$7.98 \cdot 10^{-4}$	4.65 s
`rf_pq` ($\nu = 100$)	0.105	$1.61 \cdot 10^{-4}$	0.03 s
`bt_tfidf_comp` ($\iota = 1, \nu = 1000$)	0.200	$7.44 \cdot 10^{-4}$	0.09 s
`bt_tfidf_ucmp` ($\iota = 1, \nu = 100$)	0.186	$7.09 \cdot 10^{-4}$	0.06 s
PLACES365 (CONCEPTS), YFCC100M	**precision**	**recall**	**time**
`rf_standard`	0.184	$7.48 \cdot 10^{-4}$	3.07 s
`rf_pq` ($\nu = 1000$)	0.104	$1.68 \cdot 10^{-4}$	0.05 s
`bt_tfidf_ucmp` ($\iota = 10, \nu = 1000$)	0.193	$6.62 \cdot 10^{-4}$	0.17 s
`bt_thres_ucmp` ($\iota = 5, \nu = 1000$)	0.192	$6.67 \cdot 10^{-4}$	0.10 s
`bt_thres_comp` ($\iota = 1, \nu = 100$)	0.154	$4.69 \cdot 10^{-4}$	**0.05 s**
PLACES365 (CONCEPTS), WIKIPEDIA	**precision**	**recall**	**time**
`rf_standard`	0.211	$8.07 \cdot 10^{-4}$	3.07 s
`rf_pq` ($\nu = 1000$)	0.104	$1.69 \cdot 10^{-4}$	0.05 s
`bt_tfidf_ucmp` ($\iota = 1, \nu = 500$)	0.202	$7.75 \cdot 10^{-4}$	0.05 s
`bt_thres_ucmp` ($\iota = 5, \nu = 1000$)	0.195	$7.90 \cdot 10^{-4}$	0.10 s
`bt_thres_ucmp` ($\iota = 1, \nu = 100$)	0.194	$7.67 \cdot 10^{-4}$	0.04 **s**

Figure 6 plots the effect of the choice of parameters on Blackthorn's performance. Each bar corresponds to the average of all results achieved by configurations on the given feature combinations with the given parameter (varying the others). Remarkably, the ι parameter has very little impact on precision and recall, and higher ι does not appear to translate to better semantic descriptiveness. This is in line with the human perception of visual and text information: only a few concepts describe an image; only a few topics describe the associated text. Given that higher ι directly translates into increased time per interaction round, it is overall advisable to choose low values of ι. Since the Ratio-64 representation allows easy selection of ϕ minimizing encoding errors, the encoding problems related to the previously free decimal precision parameter [45] are not an issue. Regarding ν, higher ν tends to lead to higher recall and mostly also to higher precision (certainly when going from 100 to 500). Similarly to the case of ι, higher ν leads to higher time per interaction round.

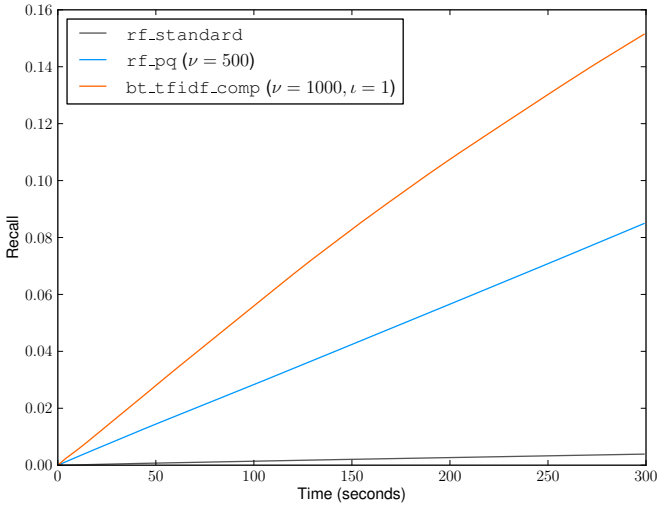

Fig. 5. Recall over time, Placing test dataset, ImageNet visual features, YFCC100M LDA topics.

TABLE III
PRECISION, RECALL AT 10 INTERACTION ROUNDS, AND TIME PER INTERACTION ROUND ON THE PLACING TEST DATASET, USING THE DENSE AVERAGE-POOLING (AP) VISUAL FEATURES.

IMAGENET (AP), YFCC100M	**precision**	**recall**	**time**
`rf_standard`	0.191	$7.29 \cdot 10^{-4}$	4.34 s
`rf_pq` ($\nu = 1000$)	0.106	$1.69 \cdot 10^{-4}$	0.05 s
`bt_thres_comp` ($\iota = 5, \nu = 1000$)	0.175	$6.87 \cdot 10^{-4}$	0.13 s
IMAGENET (AP), WIKIPEDIA	**precision**	**recall**	**time**
`rf_standard`	0.208	$9.37 \cdot 10^{-4}$	4.15 s
`rf_pq` ($\nu = 1000$)	0.106	$1.72 \cdot 10^{-4}$	0.05 s
`bt_thres_comp` ($\iota = 5, \nu = 1000$)	0.199	$8.56 \cdot 10^{-4}$	0.13 s
PLACES365 (AP), YFCC100M	**precision**	**recall**	**time**
`rf_standard`	0.172	$5.90 \cdot 10^{-4}$	4.44 s
`rf_pq` ($\nu = 1000$)	0.103	$1.64 \cdot 10^{-4}$	0.05 s
`bt_thres_comp` ($\iota = 5, \nu = 1000$)	0.150	$5.12 \cdot 10^{-4}$	0.13 s
PLACES365 (AP), WIKIPEDIA	**precision**	**recall**	**time**
`rf_standard`	0.195	$8.24 \cdot 10^{-4}$	4.35 s
`rf_pq` ($\nu = 1000$)	0.103	$1.64 \cdot 10^{-4}$	0.05 s
`bt_thres_comp` ($\iota = 5, \nu = 1000$)	0.174	$7.02 \cdot 10^{-4}$	0.13 s

Thus, ν can be viewed as a trade-off parameter. Regarding feature selection, thresholded selection slightly outperforms top-feature selection on precision and recall, while being itself outperformed by TF-IDF selection. Thresholded selection is noticeably faster than the others. This is due to it introducing additional zeros in the features and as a result encoding less information in total. Generally, it is advisable to pick TF-IDF selection for maximizing precision/recall, and thresholded selection for maximal speed. Regarding the feature variant used for SVM training, using the uncompressed features tends to have an edge over the compressed features with respect to precision and time per interaction round. Overall, while some parameters have higher impact than others, the parameter choices allow customization of Blackthorn and design choices for the speed-accuracy trade-off axis.

Table III summarizes the results of experiments with the dense visual features, i.e., the outputs of the average-pooling layer of each respective deep net. Blackthorn results are on par with `rf_standard`, despite the Ratio-64 representation design heavily favouring sparse features: Blackthorn achieves 87–96% of `rf_standard`'s precision and 85–94% of the recall, whilst maintaining the massive speed-up. Similarly to the case of sparse visual features, `rf_pq` is shown to be fast, but inferior with regard to relevance in comparison with both Blackthorn and `rf_standard`. Interestingly enough, the best Blackthorn results on each combination of features in the dense case are achieved by the same parameter configuration that suits the dense features well. Indeed, $\iota = 5$ accounts for the necessity of recording more features in the dense case compared to the sparse case favouring $\iota = 1$. The thresholded feature selection is the one emphasizing features with exceptional values the most, which is valuable when we need to discard a large number of feature values per item. Overall, whilst Blackthorn does not bring information gain when using dense visual features, we believe the answer to the fourth experimental question is that even though Blackthorn was not designed for dense features, it performs adequately on them with respect to relevance while maintaining the speed-up brought by the efficient Ratio-64 representation.

Regarding scalability, Blackthorn succeeds with respect to interactivity: on the entire YFCC100M dataset, the $\iota = 1$ configurations take **between 0.8 and 1.1 seconds** per interaction round. The performance with respect to relevance is adequate. The precision at interaction round ranges from 0.09 to 0.39. Notably, the higher values in this range surpass those on the smaller Placing test dataset. We have two explanations for this. Firstly, the Placing test dataset is specifically designed to offer a challenge in determining places around the world, unlike the general YFCC100M dataset. Secondly, the precision numbers report the performance within the 25 suggestions each round, i.e., the absolute top of the ranking corresponding to very early precision. In this case, using the entire YFCC100M dataset offers a larger pool of relevant candidates, which in the top-25 case appears to outweigh the proportionally increased size of the set of negatives. Indeed, one actor in the 100M setting corresponds to roughly 4.5 million relevant items on average, which is three times the size of the entire Placing test dataset — and we still need only the top 25, because the user's capacity to process items interactively does not scale with the dataset. The recall after 10 interaction rounds ranges from $7 \cdot 10^{-6}$ to $4 \cdot 10^{-5}$. Note that the apparent small values are caused by the sheer size of the actors. The parameter influence on the results on the YFCC100M dataset mirrors the one on the smaller dataset shown in Figure 6, with one notable exception: training on the original uncompressed features (the `ucmp` configuration) suffers from crippling feature file random access times, which increase the length of one interaction round to tens of seconds. Thus, in the large-scale case, the model should be trained on compressed features at all times. Blackthorn is much stronger than `rf_pq` on the large dataset: whilst `rf_pq` is slightly faster than Blackthorn at 0.7 seconds per interaction round, the precision and recall values of Blackthorn are much higher: `rf_pq` reaches precision of 0.04–0.05 and recall of $2 \cdot 10^{-6}$–$3 \cdot 10^{-6}$. These scalability results complement

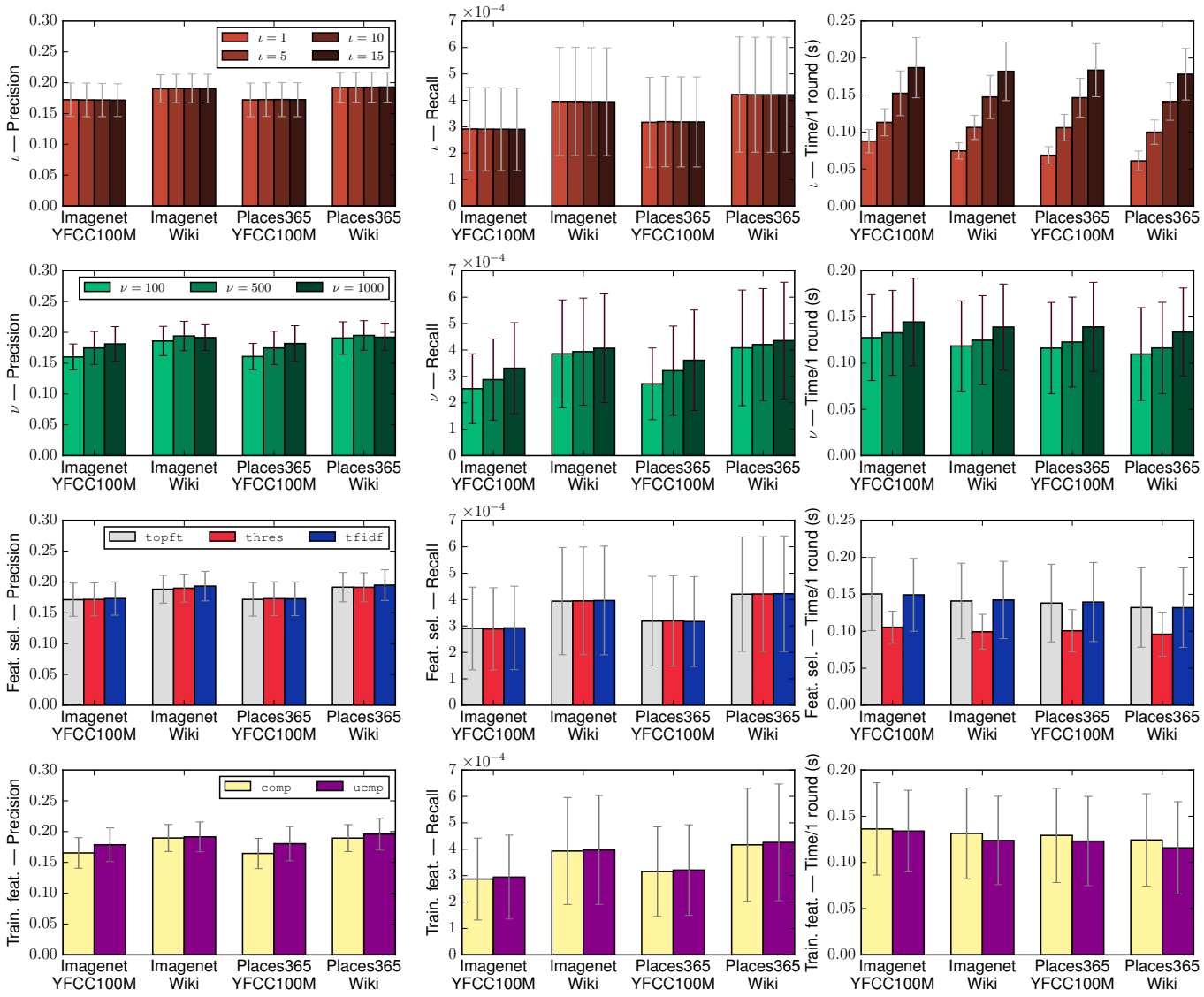

Fig. 6. The effect of parameter choice on Blackthorn results (Placing test dataset).

those reported in our previous work [45]. In this work, we have increased the tolerance for the distance from city center, focusing on generally semantically similar results, whereas the previous results without the increased tolerance correspond to a more specialized, "needle-in-the-haystack" approach. Overall, Blackthorn is shown to be scalable, as it is able to produce relevant results in truly interactive time on large-scale data.

VII. Conclusion

In this paper, we have presented Blackthorn, an efficient interactive multimodal learning framework which supports full interactive-learning-based analysis of large-scale collections with up to 100M multimedia items. The Ratio-64 compression method of Blackthorn is shown to massively reduce the size of multimodal features. In addition, it not only preserves most of the information contained in the original features, but combined with feature selection at times yields better interactive learning performance than the original features. Blackthorn yields a massive speed-up in comparison to the competing relevance feedback algorithms. The experiments further show that Blackthorn is suitable for the analysis of Web-scale datasets. It is able to learn on the fly from the user-provided training samples, and one interaction round on a collection with 100 million items takes roughly a second. Its high efficiency and low resource cost would also support multi-category exploration with proactive suggestions by the system. The analysis can be performed on a single standard high-end workstation with 64 GB RAM and 16 CPU cores. In order to foster further research, the Blackthorn software package has been made available as an open-source tool. In conclusion, Blackthorn is a step forward towards fully harnessing the wealth of information contained in large-scale multimedia collections.

ACKNOWLEDGMENT

This work is part of research projects Sort-It-Out: Visual Analytics for Multimedia collections, project number 12540, and Database Support for Multimedia Analytics, project number 040.11.525, which are (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

REFERENCES

[1] English wikipedia dump. https://dumps.wikimedia.org/enwiki/. Downloaded on 03-12-2015.

[2] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. *On the Surprising Behavior of Distance Metrics in High Dimensional Space*, pages 420–434. Springer Berlin Heidelberg, 2001.

[3] R. Aly, A. Doherty, D. Hiemstra, and A. Smeaton. *Beyond Shot Retrieval: Searching for Broadcast News Items Using Language Models of Concepts*, pages 241–252. Springer Berlin Heidelberg, 2010.

[4] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[5] J. Bernd, D. Borth, C. Carrano, J. Choi, B. Elizalde, G. Friedland, L. Gottlieb, K. Ni, R. Pearce, D. Poland, K. Ashraf, D. A. Shamma, and B. Thomee. Kickstarting the Commons: The YFCC100M and the YLI corpora. MMCommons, pages 1–6, 2015.

[6] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is 'nearest neighbor' meaningful? In *ICDT*, 1999.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[8] S. Bondugula, V. Manjunatha, L. S. Davis, and D. Doermann. Shoe: Sibling hashing with output embeddings. In *ACM MM*, pages 823–826, 2015.

[9] J. Choi, C. Hauff, O. V. Laere, and B. Thomee. The Placing task at MediaEval 2015. MediaEval, 2015.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, Sept. 1995.

[11] O. de Rooij and M. Worring. Active bucket categorization for high recall video retrieval. *IEEE TMM*, 15(4):898–907, June 2013.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[13] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik. Asymmetric distances for binary embeddings. *IEEE TPAMI*, 36(1):33–47, 2014.

[14] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *VLDB*, pages 506–515, 2000.

[15] T. Huang, C. Dagli, S. Rajaram, E. Chang, M. Mandel, G. E. Poliner, and D. Ellis. Active learning for interactive multimedia retrieval. *Proc. IEEE*, 96(4):648–667, 2008.

[16] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 33(1), 2011.

[17] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE TPAMI*, 34(9):1704–1716, 2012.

[18] L. Jiang, S.-I. Yu, D. Meng, Y. Yang, T. Mitamura, and A. G. Hauptmann. Fast and accurate content-based semantic search in 100M internet videos. In *ACM MM*, pages 49–58, 2015.

[19] Y. S. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *IEEE CVPR*, 2014.

[20] S. Kalkowski, C. Schulze, A. Dengel, and D. Borth. Real-time analysis and visualization of the YFCC100M dataset. In *MMCommons*, pages 25–30, 2015.

[21] A. Kovashka, D. Parikh, and K. Grauman. Whittlesearch: Interactive image search with relative attribute feedback. *International Journal of Computer Vision*, 115(2):185–210, 2015.

[22] M. Larson, M. Soleymani, P. Serdyukov, S. Rudinac, C. Wartena, V. Murdock, G. Friedland, R. Ordelman, and G. J. F. Jones. Automatic tagging and geotagging in video collections and communities. In *ICMR*, pages 51:1–51:8, 2011.

[23] H. Lejsek, B. T. Jónsson, and L. Amsaleg. NV-Tree: nearest neighbors at the billion scale. In *ICMR*, 2011.

[24] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu. Spectral hashing with semantically consistent graph for image indexing. *IEEE TMM*, 15(1):141–152, 2013.

[25] I. Mironică, B. Ionescu, J. Uijlings, and N. Sebe. Fisher kernel temporal variation-based relevance feedback for video retrieval. *CVIU*, 143:38 – 51, 2016.

[26] D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg. Indexing and searching 100m images with map-reduce. In *ICMR*, pages 17–24, 2013.

[27] M. Norouzi, A. Punjani, and D. Fleet. Fast search in hamming space with multi-index hashing. In *CVPR*, pages 3108–3115, 2012.

[28] C. North. Towards measuring visualization insight. *IEEE TCGA*, 26(3):6–9, 2006.

[29] V. Rao, P. Jain, and C. Jawahar. Diverse yet efficient retrieval using locality sensitive hashing. In *ICMR*, pages 189–196, 2016.

[30] B. Richard. Adaptive control processes: A guided tour, 1961.

[31] S. Rudinac, M. Larson, and A. Hanjalic. Leveraging visual concepts and query performance prediction for semantic-theme-based video retrieval. *Int J MIR*, 1(4):263–280, 2012.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[33] K. Schoeffmann. A user-centric media retrieval competition: The video browser showdown 2012-2014. *IEEE MM*, 21(4):8–13, 2014.

[34] C. G. M. Snoek, M. Worring, O. d. Rooij, K. E. A. van de Sande, R. Yan, and A. G. Hauptmann. VideOlympics: Real-time evaluation of multimedia retrieval systems. *IEEE MultiMedia*, 15(1):86–91, 2008.

[35] C. G. M. Snoek, M. Worring, and A. W. M. Smeulders. Early versus late fusion in semantic video analysis. In *ACM MM*, pages 399–402, 2005.

[36] N. Sundaram, A. Turmukhametova, N. Satish, T. Mostak, P. Indyk, S. Madden, and P. Dubey. Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. *VLDB*, 6(14):1930–1941, 2013.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

[38] B. Thomee, B. Elizalde, D. A. Shamma, K. Ni, G. Friedland, D. Poland, D. Borth, and L.-J. Li. YFCC100M: The new data in multimedia research. *Commun. ACM*, 59(2):64–73, 2016.

[39] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[40] R. Řehůřek and P. Sojka. Software framework for topic modelling with large corpora. In *LREC*, pages 45–50, 2010.

[41] H. D. Wactlar, T. Kanade, M. A. Smith, and S. M. Stevens. Intelligent access to digital video: Informedia project. *Computer*, 29(5):46–52, 1996.

[42] J. Wang, H. T. Shen, S. Yan, N. Yu, S. Li, and J. Wang. Optimized distances for binary code ranking. In *ACM MM*, pages 517–526, 2014.

[43] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proc. IEEE*, 98(6):1031–1044, 2010.

[44] E. S. Xioufis, S. Papadopoulos, Y. Kompatsiaris, G. Tsoumakas, and I. P. Vlahavas. A comprehensive study over VLAD and product quantization in large-scale image retrieval. *IEEE TMM*, 16(6), 2014.

[45] J. Zahálka, S. Rudinac, B. T. Jónsson, D. C. Koelma, and M. Worring. Interactive multimodal learning on 100 million images. In *ICMR*, pages 333–337, 2016.

[46] J. Zahálka, S. Rudinac, and M. Worring. Analytic quality: Evaluation of performance and insight in multimedia collection analysis. In *ACM MM*, pages 231–240, 2015.

[47] J. Zahálka and M. Worring. Towards interactive, intelligent, and integrated multimedia analytics. In *IEEE VAST*, pages 3–12, 2014.

[48] L. Zhang, Y. Zhang, J. Tang, X. Gu, J. Li, and Q. Tian. Topology preserving hashing for similarity search. In *ACM MM*, pages 123–132, 2013.

[49] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *arXiv*, 2016.