Efficient Mining of Repetitions in Large-Scale TV Streams with Product Quantization Hashing

Jiangbo Yuan, Guillaume Gravier, Sébastien Campion, Xiuwen Liu, Hervé Jégou

To cite this version:

Jiangbo Yuan, Guillaume Gravier, Sébastien Campion, Xiuwen Liu, Hervé Jégou. Efficient Mining of Repetitions in Large-Scale TV Streams with Product Quantization Hashing. Workshop on Web-scale Vision and Social Media, in conjunction with ECCV, Oct 2012, Firenze, Italy. hal-00731090

HAL Id: hal-00731090

https://inria.hal.science/hal-00731090

Submitted on 12 Sep 2012

Efficient Mining of Repetitions in Large-Scale TV Streams with Product Quantization Hashing

Jiangbo Yuan[1], Guillaume Gravier[2], Sébastien Campion[2], Xiuwen Liu[1], and Hervé Jégou[2]

[1] Florida State University, Tallahassee, FL 32306, USA
[2] INRIA-IRISA, 35042 Rennes Cedex, France

Abstract. Duplicates or near-duplicates mining in video sequences is of broad interest to many multimedia applications. How to design an effective and scalable system, however, is still a challenge to the community. In this paper, we present a method to detect recurrent sequences in large-scale TV streams in an unsupervised manner and with little *a priori* knowledge on the content. The method relies on a product k-means quantizer that efficiently produces hash keys adapted to the data distribution for frame descriptors. This hashing technique combined with a temporal consistency check allows the detection of meaningful repetitions in TV streams. When considering all frames (about 47 millions) of a 22-day long TV broadcast, our system detects all repetitions in 15 minutes, excluding the computation of the frame descriptors. Experimental results show that our approach is a promising way to deal with very large video databases.

1 Introduction

Mining repetitions in video data consists in finding occurrences of repeating segments, also referred to as motifs or repetitions, within the video. In the most general setup, no prior knowledge of the relevant motifs is available. A motif is solely characterized by the fact that it is repeated with very limited variability. Finding out such repeated segments in an automatic manner is of interest for several applications, in particular for TV stream structuring where repeating inter-programs—e.g., advertisements, jingles, credits—are key elements [1–3] to be identified. Motif discovery in video is also used for commercial detection [4] or to analyze news video [5]. However, most methods proposed in the literature do not scale up and can hardly deal with more than a few million frames.

In this paper, we focus on efficiently discovering unknown and repeating sequences in very long TV streams, where the targeted repetitions are either short repeating clips or more complex sequences such as commercials, trailers, or longer programs. A naive approach to the problem consists in using an off-the-shelf video retrieval system to index all the frames and retrieve repeated ones by submitting each frame in turn as a query. The efficiency of such query-based approaches is improved by using approximate nearest neighbor search techniques [6, 5], yet this approach is tractable only on a limited scale as the complexity is quadratic in the amount of data. Frame sub-sampling is usually performed to skirt the scale issue, with typically one frame extracted per shot in a repeatable manner, however at the cost of a reduced temporal precision. A better way to find all cross-neighbors consists in directly constructing a k-nearest neighbor graph. But even state-of-the-art algorithms [7, 8] are limited to a few million vectors and cannot deal with week-long TV streams, in particular because of memory requirements for such methods where a temporary version of the graph must be stored in memory during the construction phase.

As an alternative to indexing, clustering [9, 10] or hashing [11, 4, 12, 3] techniques are used to discover repeating segments by grouping similar frames. Clustering and hashing methods are close in spirit as they share the same underlying goal of mapping frame-level features to integers, based on a partition of the feature space [13]. Clustering approaches better fit the data distribution, but traditional clustering methods produce a relatively coarse partition, which is detrimental for the precision of the matching and for the complexity of the subsequent stages. In contrast, hashing-based methods enable fast projection with a reasonable large hash table. It is however difficult, if possible, to balance the recall and precision of the projection, in particular because the lack of data adaptation leads to unevenly group the vectors [13].

In this work, we propose a method for efficient discovery of repeated segments in large TV streams, bridging the gap between quantization techniques and hashing techniques. In particular, we show the interest of a hashing scheme based on *product quantization hashing* to combine the efficiency of hashing techniques to produce hash keys with the possibility to better fit the data distribution offered by clustering. The scheme is inspired by previous work on the use of product quantization (PQ) as a descriptor encoding approach for approximate nearest neighbor search in the compressed domain [14]. In our case, product quantization is used for hashing purposes to overcome the efficiency and scalability limitations of traditional clustering approaches. Compared to k-means and hierarchical k-means, product quantization is able to implicitly generate a large number of cells (as large as 2^{64} or even larger) with a very fast learning stage, and, as importantly, with a very compact representation of the centroids. As a consequence, the resulting partition is of significantly better quality than in traditional hashing-methods with comparable hash-table size.

In the context of repetition mining, product quantization hashing is used to assign each frame in a video stream to some bucket based on the resulting code with a typical size of 32 bits. As a result of the hashing step, each bucket contains visually similar frames—as defined by the representation selected for frames, GIST features in our case—from which repeated sequences are searched.

The paper is organized as follows. After describing the frame representation used, Section 2 describes the proposed product quantization hashing scheme. Section 3 describes the discovery procedure used to extract maximal-length repeated patterns from the hash table. Experimental results on three weeks of annotated TV data (47 million frames) are reported in Section 4. Finally, concluding remarks are given in Section 5.

2 Product quantization hashing

As previously mentioned, we want to consider all video frames in the problem of discovering repeated patterns so as to maintain good temporal localization. Clearly, this choice is possible only if one defines a representation of a frame that is both compact, discriminative and exhibit an efficient indexing structure. To this end, we use global descriptors for each frame. Product quantization is then used to efficiently map global descriptors to a very large number of clusters. Finally, a hash code is derived from the product quantization. We describe in turn each step of the process.

2.1 Frame representation

Similar to previous work on TV stream description [12, 3], frames are represented by color GIST descriptors [15] which are especially effective to cope with compression artifacts if an image has undergone no or limited geometrical distortion [16], as is typically the case in video. A GIST feature is a low-dimensional representation that captures the global layout of a scene, and is specifically successful at recognizing different categories of scenes that avoids the recognition of individual objects or regions. The components of the description aim at reflecting the so-called spatial envelope of the images, which is related to a set of perceptual dimensions (naturalness, openness, roughness, expansion, ruggedness) representing the dominant spatial structure of a scene.

In this work, the image is divided into a 4x4 spatial resolution grid at 3 different scales, and orientation histograms are computed in each of the scale/spatial positions. Given that the description is computed for the three RGB channels, a frame is finally described by a 960-dimensional vector representation. Finally, PCA is applied, reducing the final dimension from 960 to 64.

2.2 Product quantization

Product quantization refers to the process of separately quantizing parts of a vector. It is a compromise between a component-wise scalar quantization and a vector quantizer applied to the full vector. By encoding the subspaces independently, a product quantizer can efficiently map the input vectors to a very large number of clusters while keeping, to some extent, the advantage of a vector quantizer. Recently, a k-means version of product quantization was proposed in an approximate nearest neighbor search algorithm [14]. The structure of the quantizer allows the estimation of the Euclidean distance between two vectors directly from the quantized indexes at query time.

Here, we use the product quantization step to assign GIST-PCA vectors representing similar visual content to a unique cluster from which a compact hash code will be derived. Given a collection of D-dimensional descriptors, the product k-means clustering step works as follows:

- Divide the space into m subspaces of dimension d each ;

- ◦ Quantize each subspace with k-means assuming the same number of clusters k for each subspace ;
- ◦ Encode each vector as an index of m integers $\in [1, k]$, known as a PQ code of length $m \times \lceil \log_2 k \rceil$ bits.

In the context of repetition discovery, product quantization offers several advantages compared to traditional clustering algorithms (e.g., flat k-means, BIRCH). In particular, the quantizers operate on low dimensional subspaces and are trained with limited number of training vectors. In our case, $20,000$ vectors randomly selected from the first day of the TV stream were used to learn the codebooks. Most importantly, PQ generates a very large number of clusters with low assignment complexity. In this paper, the regular k-means was trained on one million vectors to produce K cells, resulting in a $K \times D$ assignment complexity. In comparison, PQ was trained on 20,000 vectors for $K = k^m$ clusters with an assignment complexity of $k \times D$.

In this work, product quantization is used so that frames in the same cluster are visually similar and are thus expected to be from sequentially close segments or from repeating segments. In other words, each frame in a cluster should belong to one of the occurrences of the same repeated pattern. While increasing the number of bits per code provides smaller quantization error leading in better approximate nearest neighbor search, this is not a desirable property here. Indeed, in the discovery problem as we formulate it, having all visually similar frames as part of the same cluster is a necessity. Within each cell, contrary to nearest neighbor search, we therefore target high recall rather than high precision. We therefore chose $m = 8$ and $k = 8$ or 32, i.e., 24 bits or 40 bits codes, instead of the 64 bits typically used in [14].

2.3 Hashing PQ codes

Even though the PQ codes are compact, the space of possible codes remain sparsely populated, even for real-world video sizes. In practice, they are further hashed to gain additional memory and computational efficiency. For instance, PQ-codes with $m = 8$ and $k = 32$ partitions the space into 32^8 distinct cells. This number is significantly larger than the total number of frames in a 24-hour long video stream, which comprises approximately 2×10^6 frames. The m-dimensional PQ-code is therefore subsequently hashed (with any good hashing function such as a universal hash function) and the resulting key is used within a regular hashing scheme. As a result, each frame is represented by a 32 bits signature, with visually similar frames sharing with high probability the same signature and therefore assigned to the same bucket. Obviously, as with any non reversible hashing functions, collisions might appear.

3 Discovering repetitions using temporal consistency

We exploit the structure of the hash table resulting from product quantization hashing to discover a set of repeated patterns (a.k.a. repetitions) in each bucket.

Algorithm 1 Find initial small repeated segments

```
→ RepSets ← {}
for each bucket do
    if bucket size > $b_{\min}$ then
        → append a new entry to RepSets
        → sort frames by their temporal indexes in ascending order
        for each segment of contiguous frames with length > $l_{\min}$, where contiguous
        means that the distance between two neighboring frames < $g_{\max}$ do
            → append the start and end frame of the segment to the current RepSets
            entry
        end for
    end if
end for
→ remove all entries in RepSets within only one segment
→ return RepSets
```

As discussed above, high recall was preferred to high precision. Filtering out noise from the bucket is therefore a necessity before identifying small repeated sequences. As a final stage, boundary refinement is applied before merging the small sequences to form maximal ones.

3.1 Extracting small repeated sequences from the hash table

The first step in identifying maximal length repeated sequences is to find in each bucket short repeated sequences. As the hashing scheme was designed to ensure high recall within each bucket to the expense of precision, finding such sequences in each bucket must cope in some way with outliers. This is performed by sorting frames within the bucket according to their time stamp in order to identify sequences of contiguous frames, allowing for small gaps between consecutive frames to deal with noise and outliers. As all frames in a bucket are visually similar, the short repeated sequences identified are deemed to be occurrences of the pattern.

The procedure is given in Algorithm 1 where the output, RepSets, of the algorithm is the list of all sets of repeated segments. The threshold $b_{\min}$ is used to avoid searching in very small buckets while $l_{\min}$ filters out very short repetitions. Finally, $g_{\max}$ is the maximum temporal gap tolerated for two frames to be considered contiguous. The use of $g_{\max}$ mostly reflects that the recall in each bucket is not perfect even if good.

The size of the RepSets list usually is considerably large: In our experiments, with $b_{\min} = 8$, $l_{\min} = 4$ and $g_{\max} = 25$ (i.e., 1 s), between 20,000 to 40,000 sets where found for a 24 hours long TV stream. This number is reduced to a few thousands after removing all singletons. However, these short repeated segments detected by the initial stage of the algorithm are far from the maximal length repeated sequences one wants to discover. The next steps therefore aim at extending each individual segment and refining their boundaries before merging related sets to form longer repeated sequences.

Algorithm 2 Segment boundary optimization

```
for each segment $S_i \in$ RepSets do
    → find the monochrome segment $M_k \in$ right before $S_i$
    if start_ID($S_i$) − end_ID($M_k$) < $b_{max}$ then
        → start_ID($S_i$) ← end_ID($M_k$)
    end if
    → find the monochrome segment $M_l \in$ right after $S_i$
    if start_ID($M_l$) − end_ID( $S_i$) < $b_{max}$ then
        → end_ID($S_i$) ← start_ID($M_l$)
    end if
    → label $S_i$ with <$M_k$, $M_l$>
end for
```

3.2 Segment boundary optimization

In the particular case of advertisements in TV streams, one can take advantage of specific markers such as inter-commercial monochrome segments as in [4, 17]. Inter-commercial markers are usually very short sequences, i.e., less than 1 s, which consist of monochrome frames with no audio signal. A large majority of the repeated sequences in TV streams corresponds to advertisements which intuitively provide an appealing feature for TV stream structuring [10]. Moreover, taking benefit from product quantization hashing, inter-commercial markers can be detected at almost no extra cost. These facts justify that specific processing be devoted to inter-commercial monochrome frames to adjust boundaries of short repeating segments.

As inter-commercial monochrome short segments are numerous in TV streams and are by definition visually very similar, they all fall into the same bucket after product quantization hashing and therefore generate an entry in the RepSets list easily identifiable by its many occurrences. The set of monochrome repeated segments is therefore characterized and identified by its abnormally large population.

Based on the location of monochrome segments, one can refine the boundaries of the repeated segments: Whenever a monochrome segment appears near the beginning (respectively, the end) of an occurrence of a repeated segment, the starting (respectively, ending) frame ID of the latter is modified to match the corresponding boundary in the monochrome segment. This procedure is formally described by Algorithm 2, where b_{max} is set to 200. The use of a threshold reflects the fact that monochrome segments appear with commercials, thus reducing the probability of optimizing the boundaries of a non commercial repeated segment.

3.3 Finding out maximal length repeated segments

The previous steps enable to identify short repeated segments which are then merged to yield longer repeated segments. For sake of comparison with other methods, the fusion stage was limited to basic operations to avoid over-fitting problems and application specific solutions. The fusion successively takes place

Table 1. Datasets and statistics of the reference repeated segments

Name	#frames	length	#rep seg
TV-DAY1	2,160,026	24 h	133
TV-DAY2	2,159,920	24 h	148
TV-22	47,517,004	528 h	1,502

at two levels. An inner-set fusion is first performed independently within each entry of the RepSets list before merging across all entries.

Inner-set fusion indicates that, in a list of short repeated segments, several may belong to a continuous program but somehow are cut by the detection. For each entry in the RepSets list, two segments with a gap (say, S_i $S_\mathrm{i+1}$) less than a threshold $g^{'}_{\mathrm{max}} = 1,000$ are merged. When we take advantage of monochrome segments, we increase $g^{'}_{\mathrm{max}} = 10,000$ if the two segments have the same labels (say, $<M_\mathrm{k}, M_\mathrm{k+1}>$, as defined in Algorithm 2).

Inter-set fusion is expensive since it has to exhaustively check all identical sets. The basic idea is the following: Given two entries r_1 and r_2 in RepSets (assume the size of r_1 is larger than the size of r_2), we mix and merge their segments together by using a gap threshold $g^{''}_{\mathrm{max}} = 500$, then we accept the returned set if its size is equal or smaller than r_2. All the accepted new repSets form the final RepSets that will be evaluated.

4 Experimental results

4.1 Dataset and ground-truth

Experiments are carried out on a 22-day long TV stream. The dataset was manually annotated using the individual TV programs as basic units. Each unit is identified by its start and end times, a title and a genre (e.g., program, jingle). A ground truth reference is built from these annotations by considering all units with the same title as repeated segments/events. Note that such repetitions vary in genre, in length and in the number of occurrences.

Experiments are carried out either on the entire dataset or on small portions identified as TV-DAY1 and TV-DAY2 (see Tab. 1), corresponding to respectively the first and second day of the stream. The former is used to train the product quantizers and to tune parameters while the latter is used for sake of comparison to existing approaches which cannot scale to the 22 days.

All the algorithms have been developed in C and Matlab (for and only for the detection part). Experiments have been performed on a single core machine under Unix.

4.2 Evaluation measures

Performance measures were designed to assess the detection of repeated segments both in terms of purity and temporal localization. Given the detected list of repeated segments—i.e., RepSets—and the ground truth list, there is often a many-to-one mapping between the lists because of the difficult sets fusion. We therefore map each entry in the ground truth list to the best corresponding entry in RepSets. Each detected segment is then mapped to at most one occurrence of the corresponding event in the ground truth, assuming two segments coincide if they overlap by more than $\theta \times l$ where l is the length of the longer segment and θ a user defined parameter. Relying on the mapping above, the detection is evaluated by the F1-score for a particular repeating segment, i.e., the harmonic mean of precision and recall defined as

- recall=(# correctly detected reference segments)/(# of reference segments)
- precision=(# correctly detected segments)/(# of detected segments)

The measurements are averaged over all repeating events to yield global measures denoted as AF (for a single-day data) and mAF (for the 22-day data).

4.3 Results

Experiments are performed with different hashing and clustering techniques, namely: standard k-means (KM), hashing (STDHASH), locality sensitive hashing (LSH), and our product quantization hashing (PQH) with 24 or 40 bits keys. Comparative results are reported in Fig. 1 for the TV-DAY2 subset for varying values of the overlap threshold θ. Final results on the entire corpus are given in Fig. 2 for PQ hashing and LSH which are the only scalable methods. In both cases, results are reported with and without boundary optimization using monochrome frames.

Standard k-means and hashing are special cases of our method: Setting $m = 1$ results in a standard k-means while defining $m = 64$ and $k = 2$ (or a small value) yields an approach very similar to standard hashing schemes as used in [11, 4, 12, 3]. LSH use random projections with a hash function defined as

$$h_{\mathrm{r}}(x) = \begin{cases} 1, \text{ if } r^{\mathrm{T}}x \geqslant 0 \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

where r is a random vector subject to a $\mathcal{N}(0, I)$ distribution. See [18] for more details. The length of binary codes (b) depends on how many hash functions are used. From the results in Fig. 1, we observe that longer codes (e.g., LSH-128) are less adapted as they allow very limited frame distortion.

Fig. 1 shows that the standard hashing method, though the fastest, provides poor results. On the opposite, the methods based on regular k-means clustering are effective, especially for large values of k. However, the learning and assignment steps are too expensive for real-time purpose, and even infeasible for the very large values of k needed for the purpose of hashing because of the lack of sufficient amount of training data. Therefore, these two methods are not adapted to process large-scale streams. In contrast, PQH-24, PQH-40, and LSH-64 provide overall competitive results with reasonable processing time. PQH-40 achieved the best timing performance (about 20 s for encoding and hashing, and 17 s for

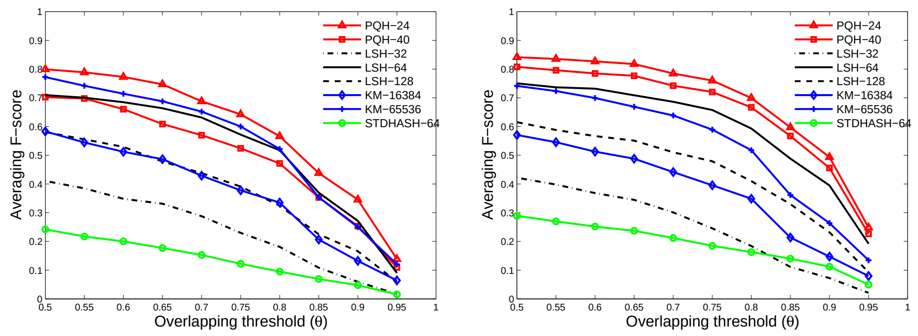

Fig. 1. Detection performance on TV-DAY2 without (left) and with (right) monochrome segments for boundary refinement.

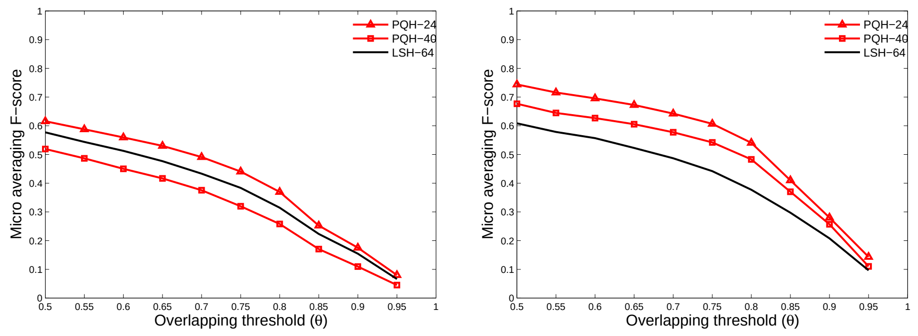

Fig. 2. Detection performance on TV-22 without (left) and with (right) monochrome segments for boundary refinement.

detection). The accuracy of PQH-24 is slightly better however with a slower processing time, in particular in the detection step (9 s for encoding and hashing, and 42 s for detection). LSH-64 is faster than PQH-24 on encoding but takes about 230 s for detection. Note that the timing performance highly depends on the size of the candidate RepSets.

Performance measures on the 22 days long TV stream, as reported in Fig. 2, demonstrate the effectiveness of our approach to accurately discover repeating segments in very large data streams.

5 Conclusion

In this paper, we have proposed a new method for efficiently and effectively detecting repeated sequences in large TV streams, without requiring prior knowledge. By hashing PQ-codes to produce compact but effective frame-signatures, our method is scalable in terms of both storage and complexity. Several directions can be envisioned to further refine the method. Boundary refinement using shot boundaries might be considered when monochrome frames are not available. Using key-frames is an option to drastically speed-up the process that needs to be explored. Finally, the PQ hashing scheme proposed meets the needs of other applications. For instance, we plan to extend the work for web-based near duplicate video detection by encoding entire video clips in order to fetch a higher-level signature for video retrieval/mining.

Acknowledgments. This project was partially supported by the Quaero project, funded by OSEO, french agency for innovation.

References

1. Naturel, X., Gros, P.: Detecting repeats for video structuring. Multimedia Tools and Applications **38** (2007) 233–252
2. Manson, G., Berrani, S.A.: Automatic tv broadcast structuring. International Journal of Digital Multimedia Broadcasting **2010** (2010) 16 pages
3. Ibrahim, Z.A.A., Gros, P.: Tv stream structuring. ISRN Signal Processing **2011** (2011) 17 pages
4. Gauch, J.M., Shivadas, A.: Finding and identifying unknown commercials using repeated video sequence detection. Computer Vision and Image Understanding **103** (2006) 80–88
5. Yang, X., Tian, Q., Member, S., Xue, P.: Efficient short video repeat identification with application to news video structure analysis. IEEE Transactions on Multimedia **9** (2007) 600–609
6. Yuan, J., Wang, W., Meng, J., Wu, Y., Li, D.: Mining repetitive clips through finding continuous paths. In: ACM. MM '07, ACM (2007) 289–292
7. Dong, W., Charikar, M., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: WWW. (2011)
8. Wang, J., Wang, J., Zeng, G., Tu, Z., Li, S.: Scalable k-nn graph construction for visual descriptors. In: Conf. on Vision and Pattern Recognition. (2012)
9. Goh, K.S.: Audio-visual event detection based on mining of semantic audio-visual labels. Proceedings of SPIE **5307** (2003) 292–299
10. Berrani, S., Manson, G., Lechat, P.: A non-supervised approach for repeated sequence detection in tv broadcast streams. Signal Processing Image Communication **23** (2008) 525–537
11. Pua, K.M., Gauch, J.M., Gauch, S.E., Miadowicz, J.Z.: Real time repeated video sequence identification. Computer Vision and Image Understanding **93** (2004) 310–327
12. Döhring, I., Lienhart, R.: Mining tv broadcasts for recurring video sequences. In: ACM. CIVR '09, ACM (2009) 28:1–28:8
13. L. Paulevé, H.J., Amsaleg, L.: Locality sensitive hashing: a comparison of hash function types and querying mechanisms. Pattern Recognition Letters (2010)
14. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. PAMI **33** (2011) 117–128
15. Oliva, A., Torralba, A.B.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV **42** (2001) 145–175
16. Douze, M., Jégou, H., Singh, H., Amsaleg, L., Schmid, C.: Evaluation of GIST descriptors for web-scale image search. In: CIVR. (2009)
17. Naturel, X., Gravier, G., Gros, P.: Fast structuring of large television streams using program guides. In: AMR: user, context, and feedback. AMR'06, Berlin, Heidelberg, Springer-Verlag (2007) 222–231
18. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: In Proc. of 34th STOC, ACM (2002) 380–388