MINING ARBITRARY-LENGTH REPEATED PATTERNS IN TELEVISION BROADCAST

Sen-ching S. Cheung

ECE Department, University of Kentucky
cheung@engr.uky.edu

Thinh P. Nguyen

ECE Department, Oregon State University
thinhq@eecs.oregonstate.edu

ABSTRACT

Mining repeated patterns in television broadcast is important to advertisers in tracking a large number of television commercials. It can also benefit long-term archival of television because historically significant events are usually marked by repeated airing of the same video clips or sound-bytes. In this paper, we describe a system that can efficiently mine repeated patterns of arbitrary lengths from television broadcast. Compared with existing work, our system has two main innovations: first, our system is robust against minor temporal variations among repeated patterns. This is important as broadcasters often perform temporal editing on commercials so as to fit them into different time slots. Second, our system does not rely on any temporal segmentation algorithm, which may lead to over- or under-segmentation of important patterns. Instead, our system scans the television broadcast with a fixed-size sliding window, summarizes each window into a hash value, and maintains a running frequency count and a reference time-stamp on each hash value. The boundaries of a repeated pattern are identified by the changes in frequency counts and reference time-stamps. Initial experiments show that our system is very efficient in identifying all the repeated commercials from 12 hours of television broadcast.

1. INTRODUCTION

Radio and television broadcasting has been a major influence in shaping the political, social, cultural, and economic trends of the twentieth century. Important events, like the attack of the world trade center or the tsunami in south east Asia are usually marked by repeated airings of the same video clips. The broadcast frequency of a particular video clip is perhaps one of the most important indicators of its historical significance. As such, the capability of finding video or audio material with high broadcast frequency can help greatly to identify important footages for news alert and preservation. Nevertheless, not all repeated broadcast are historically significant. Perhaps the largest category of repeated broadcast belongs to television commercials. Even though finding repeated commercials may not be important to historians, it is crucial for advertisers to track the air time of their commercials and for companies to monitor new products from their competitors.

The aforementioned applications will benefit greatly from a system that can continuously monitor a large number of broadcast channels, and identify, in *real-time*, the *duration*, the *frequency*, and the *location* (time and channel) of every repeated pattern. We call this the *repeated-pattern mining* problem. Due to the nature of the target content like news footages and tv commercials, their presence are usually not indexed in the Electronic Programming Guide (EPG). Thus, we contend that the only viable approach is to use the content-based approach – identifying a multimedia item by extracting salient audiovisual features directly from the content. However, building such a system poses many technical challenges, mainly due to the enormous volume of broadcast information, its continuous nature, and the requirement of a real-time response.

In this paper, we describe a system that can efficiently mine repeated patterns of arbitrary lengths from television broadcast. Compared with existing work, our system has two main innovations: first, our system is robust against minor temporal variations among repeated patterns. This is important as broadcasters often perform temporal editing on commercials so as to fit them into different time slots. Our system accommodates such variations by using a randomized summarization technique called ViSig [1]. The advantage of ViSig is that it can be used to match patterns that are only partially overlapped. Second, our system does not rely on any temporal segmentation algorithm, which may lead to over- or under-segmentation of important patterns. Instead, our system scans the television broadcast with a fixed-size sliding window, summarizes each window into a hash value using ViSig, and maintains a running frequency count and reference time-stamp for each hash value. We will show that the boundaries of a repeated pattern can be easily identified by the changes in frequency counts and reference time-stamps.

The paper is organized as follows: we first review related work and motivate our approach in Section 2. We then describe our proposed algorithm in Section 3 and present experimental results in 4. We conclude by discussing ongoing work in Section 5.

2. RELATED WORK

The problem of mining repeated patterns in a time series is important in many applications such as discovering similarities in genomic data [2] and identifying themes in music [3]. Efficient algorithms exist for mining repeated patterns in such applications [4]. The basic idea is to build a suffix tree that stores all the possible subsequences in a compact form. The algorithm described in [4] is an offline algorithm with $O(N \log N)$ time complexity and $O(N)$ memory requirement where N is the length of the time series.

Suffix trees are appropriate for genomic data and music because each symbol belongs to a limited set of alphabets. A typical video sequence, on the other hand, requires a high-dimensional feature vector to represent each video frame. This renders suffix trees useless as the alphabet size is effectively infinite. The high dimension, however, is in fact a blessing in disguise because the probability of observing two identical feature vectors in different time is so rare that their occurrences strongly indicate the presence of repeated patterns. Furthermore, as the broadcast video is usually of good quality, with an appropriate feature design, video frames with the same content are likely to produce not similar but *identical* feature vectors. Thus, it is possible to use a simple table lookup to mine most of the repeated video frames [5, 6].

Nevertheless, the presence of the same video frame does not always lead to the discovery of meaningful repeated patterns. Black frames, frames with station logos, or even frames with a news anchor may occur repeatedly at different time but the contexts to

which they belong may be completely different. Thus, one has to consider a longer segment of video in mining repeated patterns. Comparing long video clips is difficult and one approach is to use feature vectors of even higher dimension to represent them. Unlike the frame-based approach, two long video clips with identical content are far less likely to produce identical feature vectors. As a result, one needs to resort to high-dimensional similarity search, which is the approach used in [7, 8]. To expedite the similarity search, sophisticated index structures are usually needed and created off-line to support fast search. Such an approach is not suitable for on-line broadcast monitoring as the index structure needs constant updating and the search response may not satisfy the real time constraint. A compromised approach called ViSig to represent long video clips was proposed in [1]. Instead of using an aggregate feature vector to represent an entire clip, ViSig uses a small subset of frames sampled from the sequence. We have chosen ViSig for our application and we will provide a brief review in Section 3.1.

Another problem that has not attracted much attention is how to mine *maximal* repeated patterns – the repeated patterns of maximal lengths that cannot be further extended. A maximal repeated pattern represents the most meaningful unit of repeated patterns. For example, it may represent a complete commercial or a music video. Most existing research assumes either a fixed time window as in [7] or uses a shot detection algorithm to define the basic unit for repeated pattern mining [6]. Besides the reliability issue of the shot detection algorithm, a maximal repeated patterns may contain a large number of shots and an efficient data structure is needed to connect them together to form the entire pattern. In this paper, we describe how maximal repeated patterns can be mined by simply using frequency counts and reference time-stamps.

3. PROPOSED ALGORITHM

3.1. Review of ViSig

We begin with a brief review of ViSig algorithm[1]. We assume each frame in a video X is represented by a feature vector x in a metric space F with distance $d(\cdot,\cdot)$. ViSig summarizes a video sequence as a tuple of n vectors called video signature. A video signature X_S of a video X is defined as follows:

$$X_S = (g_X(s_1), g_X(s_2), \ldots, g_X(s_n)) \quad (1)$$

where $g_X(s) = \arg\min_{x \in X} d(x, s)$ and $S = \{s_1, s_2, \ldots s_n\}$ is a fixed set of n random seed vectors. In our application, X represents all the frames within a sliding window. Notice that in computing the signatures of successive sliding windows, there is no need to recompute the distances between all the feature vectors in a window and the random seed vectors. Assume the current window cover feature vectors x_{t-W+1} to x_t. For each seed vector s_i, we maintain a sorted list L_i of distances between every feature vector within the window and s_i. When the window advances and a new feature vector x_{t+1} arrives, we first discard the oldest distance $d(x_{t-W+1}, s_i)$ from L_i and insert the new distance $d(x_{t+1}, s_i)$ to L_i while maintaining its order. The complexity of this process is $O(\log W)$. The signature vector for this new window is simply the one corresponding to the smallest distance at the top of L_i.

To compare two video signatures, we define the signature similarity as the fraction of similar signature vectors as follows:

$$\mathrm{Sim}(X_S, Y_S) = \sum_{i=1}^{n} \mathbf{1}\left(d(g_X(s_i), g_Y(s_i)) \le \epsilon\right)/n \quad (2)$$

where $\mathbf{1}(\cdot)$ is one if the predicate inside is true and zero otherwise and $\epsilon > 0$ defines the similarity criterion between vectors. We declare two video sequences as similar if their signature similarity is larger than a user-defined threshold. In [1], we have shown that $\mathrm{Sim}(X_S, Y_S)$ is an unbiased estimate of the volume of the intersection between the Voronoi diagrams created by X and Y. It can be shown that the size of the signature n depends on the desired measurement accuracy, but not on the length of the video sequence. Another result from [1] shows that for a given video X, there exist some seed vectors that are more "robust" than the others in identifying video sequences similar to X – for any video X' highly similar to X, the probability of $d(g_X(s), g_{X'}(s)) \le \epsilon$ for some $s \in S$ may be significantly higher than the other seed vectors. Geometrically, a seed vector is robust if it is far away from the Voronoi cell boundary. Later in Section 3.2, we will take advantage of this fact in simplifying our design.

3.2. Frequency Counts and Reference Time-stamps

The basic idea of using frequency counts and reference time-stamps for mining repeated pattern is straightforward. Consider the process of scanning the video stream with a sliding window that contains W vectors. Assume all the feature vectors within the sliding window $[t-W, t]$ are hashed into a symbol h_t. For every sliding window, we update a hash table that contains a tuple (`freq_count`, `ref_time`), indexed by all the possible symbols. Let `ref_time`(t) indicate the time when the symbol h_t first appears, and `freq_count`(t) indicate the number of times the symbol h_t has appeared. As adjacent windows are likely to produce the same symbol, we only update `freq_count` once for a continuous stream of identical symbols.

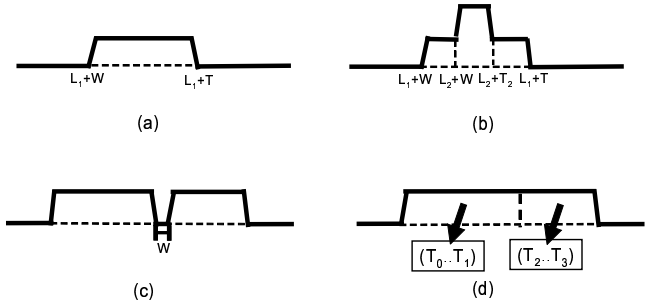

Fig. 1. *Sample* `freq_count` *sequences.*

Consider a pattern that first occurs between time L_0 and $L_0 + T$, and repeats again between time L_1 and $L_1 + T$. Assume that no portion of this pattern has appeared elsewhere, the `freq_count` of every sliding-window symbol h_t for $t \in [L_1 + W, L_1 + T]$ must be two. This is illustrated in Figure 1(a). By using the rising and falling edges of `freq_count`(t), we can easily identify the boundaries of the repeated pattern. If a part of the pattern, say between L_2 and $L_2 + T_2$, has appeared elsewhere before L_1, then `freq_count`(t) will be similar to Figure 1(b). The repeated patterns in this case can be easily handled by using a simple stack. When encountering a rising edge at time t, we create a new pattern, set the beginning time to $t - W$, and push it in the *repeated pattern stack*. When we encounter a falling edge at a later time t', we pop all the patterns from the stack whose frequency counts are larger than the current `freq_count`(t'), set the ending time of all these patterns to t', and output these patterns.

The situation becomes slightly trickier when there are two distinct patterns that are adjacent to each other and have the same frequency count. The perfect scenario is that the sliding windows at

the transition produce zero counts like in Figure 1(c) and we can separate the two patterns completely. As we plan to use ViSig to summarize each window, it is possible that the transition may disappear as ViSig is a sampling technique. In such case, we utilize the reference time-stamp sequence `ref_time`(t) to identify the boundary: if there is a significant discontinuity in the `ref_time`, as shown in Figure 1(d), we pop the top pattern from the stack and create a new pattern starting at the same time. The `ref_time` can also be used to establish correspondence among repetitions of the same pattern as they share the same range of `ref_time` time-stamps.

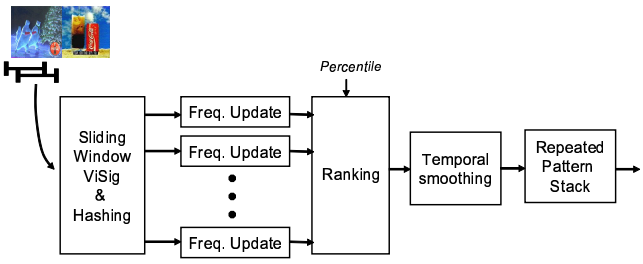

Fig. 2. *Proposed system for repeated pattern mining.*

We are now able to describe our proposed system for mining repeated patterns. The overall architecture is shown in Figure 2. For each sliding window, we compute a n-vector video signature based on the sliding-window ViSig described in Section 3.1. We declare two sliding windows X and Y to be the same if p out of the n vectors in their signatures are identical, i.e. $\mathrm{Sim}(X_S, Y_S) \geq p/n$ where $\epsilon = 0$ as defined in (2).

To account for all possible choices of the p vectors, we will need as many as $C(n,p) = O(n^p)$ hash tables. This is too computational demanding unless both n and p are very small. Fortunately, as pointed out earlier in Section 3.1, repeated patterns usually have the same subset of similar signature vectors, which correspond to those robust seed vectors that are far away from the Voronoi cell boundary. Thus a simple heuristics is to use one hash table for each signature vector to record its frequency count and reference time-stamp, which is the function of the *Freq. Update* component in Figure 2. For a given signature X_S, those hash tables corresponding to the robust seed vectors will give the correct `freq_count`, while the other hash tables are likely to underestimate the count. Thus, to combine the frequency counts from all the hash tables, we rank them in increasing order and choose the $\alpha = 1 - p/n$ percentile as the output, denoted as `freq_count1`. This will produce the same frequency count sequence as the optimal version, provided that there are at least p robust vectors for each signature. The corresponding `ref_time1` is chosen to be the oldest time-stamps among all the signature vectors that share the same frequency count as `freq_count1`. Such a strategy makes it more likely for all repetitions of the same pattern to share the same reference time-stamps. The process of combining results from multiple hash tables to produce `freq_count1` and `ref_time1` is performed by the *Ranking* component in Figure 2.

The *Temporal smoothing* component in Figure 2 is to eliminate short bursts of erroneous frequency counts due to noisy data or the presence of common but unimportant repeated patterns like black frames or station logos. This component holds a temporal buffer of `freq_count1` and `ref_time1`, and outputs `freq_count2` that corresponds to the most common frequency counts in the buffer, and `ref_time2` that corresponds to the median of all reference time-stamps that share the same frequency count as `freq_count2`. `freq_count2` and `ref_time2` are then passed to the *Repeated pattern stack*, whose function has already been described earlier. The final output is the list of repeated patterns identified by the proposed algorithm.

4. EXPERIMENTS

To show the retrieval performance of our proposed algorithm, we have captured two 6-hour long continuous broadcast from two television channels "Comedy Central" and "Turner Broadcasting System (TBS)". The primary programming in these channels are movies, drama and comedy shows. These programs are captured from analog cable and compressed into MPEG-1 bitstreams. All repeated patterns are commercials and they are manually identified to form ground-truth clusters of repeated patterns. We use two sets of ground-truth: set A consists of repeated patterns that are identical, while set B include all patterns that are "approximately identical" – some of them are edited versions of the others, while others are commercials that are shown in a smaller size along with the rolling credits from the previous programs. The details of the ground-truth sets are as follows:

Set	no. clusters	Avg. size	Avg. length
Comedy (A)	43	2.40	30.97 sec
Comedy (B)	51	2.57	29.96 sec
TBS (A)	25	2.88	26.57 sec
TBS (B)	26	2.96	26.96 sec

Table 1. *Statistics of repeated commercials in test videos*

In our experiments, we use ordinal features on both the luma and chroma color channels [9]. The luma feature is the ranking of average intensities of the 4×4 partition of the Y frame. The chroma feature is the concatenation of the two rankings of the 3×3 partitions of the Cb and Cr frames. l_1 distance is used in computing signatures for these features. The two features are combined in such a way that there is a match in the sliding window if either feature finds it to be a match. Thus, in the implementation, separate signatures are generated for the two features and the results are combined in the *Ranking* component in Figure 2: whichever produces the higher frequency count or older reference time if the frequency counts are equal will be passed on to the next step. A sliding window of 5 seconds is used and each window is summarized by 32 signature vectors. Each signature vector is hashed into a 32-bit number based on the algorithm in [10]. Instead of a full hash table with 2^{32} entries, the frequency counts and reference time-stamps are stored as a B-tree indexed by the hash values of the features. To accommodate small variations in feature vectors, in the *Freq. Update* step in Figure 2, we consider the entries corresponding to the hash values of the signature vector and all of its perturbed versions. The perturbation is limited to those within l_1 distance of two, i.e. the exchange of neighboring ranks. Thus, a luma feature vector has 15 perturbed versions and a chroma feature vector has $8 + 8 = 16$ perturbed versions. Again, whichever versions produce the highest frequency count, or the oldest reference time-stamp in case of equal count will be passed on to the next step. As all the repeated patterns in the ground-truth set are longer than 10 seconds, we use a temporal smoothing buffer of 20 seconds and discard any identified repeated patterns that are shorter than 10 seconds.

To compare the output set of repeated patterns with the ground-truth, we first measure the amount of overlap in time between all pairs of patterns from the two sets. The amount of overlap is defined as the average of two ratios: one between the length of the

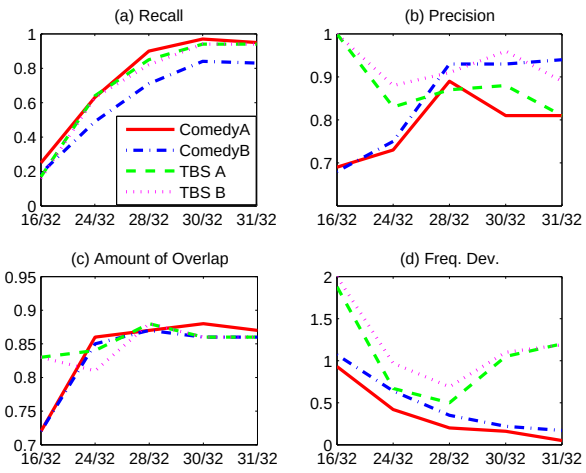

Fig. 3. *Retrieval performances of the proposed system.*

intersection and the ground-truth pattern, and the other between the intersection and the output pattern. After sorting all the overlapped pairs in the decreasing order of their amount of overlap, we scan the sorted list and declare a pair as a match if neither patterns in the pair has been matched with other patterns before. We measure "recall" and "precision" based on the ratios between the number of matched patterns and the total number of patterns in the ground-truth set and the output set respectively. We also measure the average amount of overlap and the deviation of the frequency counts among all the matched patterns. All the measurements are performed for five different percentile levels: 16/32, 24/32, 28/32, 30/32, and 31/32. Note that higher percentile level implies a less stringent requirement in declaring two signatures as similar. Recall, precision, average amount of overlap, and average deviation of the frequency counts as functions of the percentile levels are shown in Figure 3 (a)–(d).

Unsurprisingly, as shown in Figure 3(a), recall improves as the percentile level for matching increases. All but "Comedy B" reach a recall level higher than 95% at high percentile levels. "Comedy B" has a number of very difficult patterns, which include commercials shown simultaneously with rolling credits of different movies. Even the commercials themselves are identical, the rolling credits are not, making spatial features like the ordinal feature unsuitable to identify them. In Figure 3(b), the precision level reaches beyond 90% for the two B sets at high percentile levels. They are higher than the corresponding A sets because ViSig is capable of identifying the approximated repeated patterns that are not present in the two A sets. The two "Comedy" curves show an unusual pattern of increase in precision when the percentile level increases. This is because some of the longer patterns are broken into multiple shorter ones due to the more stringent matching criterion. As a result, there are multiple output patterns corresponding to the same ground-truth pattern. Since only one of them can match with the ground-truth, the precision value decreases. The average amount of overlap, shown in Figure 3(c), stays mostly at the 85% level, except at the lowest percentile level. This is reasonable as the average length of repeated patterns is 30 second, a sliding window of 5 second may lead up to 5/30 = 17 % of error in estimating the boundaries of the patterns. For the last plot of the deviation in frequency counts, once again the two sequences have quite different behavior. While the system makes close to zero mistake in "Comedy" at the high percentile levels, it is off by more than one for "TBS". A closer examination reveals that the estimates are correct for most patterns, but there are differences as big as 15 for a small number of repeated patterns in TBS. It turns out that they are all commercials about a movie TBS is running that day and a segment of that movie is embedded in many variations of the station's promotional commercials. While these variations belong to different ground-truth clusters, our system identifies them as the same repeated pattern, making the frequency count much higher than expected.

5. CONCLUSIONS

In this paper, we have described a system to find repeated patterns from television broadcast. We have introduced an novel idea of using frequency counts and reference time-stamps of sliding windows in identifying arbitrary-length repeated patterns. The sliding windows are summarized by the ViSig method which has the advantage of capturing approximately similar patterns. Preliminary results on two 6-hour broadcast have been presented. There are a number of obvious improvements that we are currently investigating. First, when applying our system to television broadcast, we need to implement an expiration policy such that entries older than a certain time are removed from the hash tables to make room for newer ones. Techniques such as those described in [11] may be applicable to reduce memory usage if we are only interested in keeping only the most frequent patterns. Second, better features are needed to further improve the recall performance. Third, more diverse material are needed to further test the robustness of the system. For example, our current system may have a hard time handling video from a 24-hour news channel as the anchor shots are all similar even though the underlying stories are completely different.

6. REFERENCES

[1] S.-C. Cheung and A. Zakhor, "Efficient video similarity measurement with video signature," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 59–74, Jan. 2003.

[2] J.T.-L. Wang, G.-W. Chirn, T.G. Marr, B. Shapiro, D. Shasha, and K. Zhang, "Combinatorial pattern discovery for scientific data: some preliminary results," in *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1994, vol. 23, pp. 115–125.

[3] D. Meredith, K. Lemstrom, and G.A. Wiggins, "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *Journal of New Music Research*, vol. 31, no. 4, pp. 321–345, Dec 2002.

[4] M. Crochemore, "An optimal algorithm for computing the repetitions in a word," *Information Processing Letters*, vol. 12, no. 8, pp. 244–250, Oct 1981.

[5] J. Oostveen, T. Kalker, and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *Recent Advances in Visual Information Systems*, Berlin, Germany, 2002, vol. 2314 of *Lecture Notes in Computer Science*, pp. 117–128.

[6] K.M. Pua, J.M. Gauch, S.E. Gauch, and J.Z. Miadowicz, "Real time repeated video sequence identification," *Computer Vision and Image Understanding*, vol. 93, pp. 310–327, 2004.

[7] Q. Tian C. Xu J. Yuan, L.-Y. Duan, "Fast and robust short video clip search using an index structure," in *ACM Multimedia's Multimedia Information Retrieval Workshop*, 2004.

[8] A. Joly, C. Frelicot, and Olivier Buisson, "Robust content-based video copy identification in a large reference database," in *Proceedings of International Conference on Image and Video Retrieval*, 2003, vol. 2728 of *Lecture Notes in Computer Science*, pp. 398–407.

[9] R. Mohan, "Video sequence matching," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, WA, May 1998, vol. 6, pp. 3697–3700.

[10] Bob Jenkins, "Algorithm alley: Hash functions," *Dr. Dobb's Journal*, Sept. 1997.

[11] G.S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *Proceedings of the 28th VLDB conference*, 2002.