Semantic analysis of a large-scale news video archive

Ichiro Ide [†,‡] **Kazuhiro Noda** [† *] **Akira Ogawa** [†] **Shin'ichi Satoh** [‡] **Hiroshi Murase** [†]

† Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

{ide, murase}@is.nagoya-u.ac.jp, {knoda, aogawa}@murase.m.is.nagoya-u.ac.jp

‡ National Institute of Informatics, Research Organization of Information and Systems
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

{ide, satoh}@nii.ac.jp

Abstract

In this paper, we introduce our recent works on semantic analysis on a large volume of news video data archived for more than five years, equivalent to approximately 900 hours of MPEG-1 video data. After briefly introducing the archive, two works that analyze the news contents based on text and image information are introduced; topic threading and cross-lingual news story retrieval. The paper introduces at the end, an important image processing method for the semantic analysis; fast near-duplicate video segment detection. Some of the works introduced in the paper are still in a preliminary stage, but we believe that they should play an important role in handling the growing amount of video contents in the future.

Keywords: Cross-lingual video retrieval, topic threading, near-duplicate video segment detection

1 Introduction

Recent advance in data storage technologies has provided us with the ability to archive many hours of video streams accessible as online digital data. Thanks to this trend, we have been able to archive a daily Japanese news show for the past five years, equivalent to approximately 900 hours of video data. Figure 1 shows the configuration of the archive, while Tab. 1 shows the specifications of its contents.

In order to make use of the contents of the archive, it is essential to provide the users with the ability to search, browse and understand the video data based on the semantic contents. Under the awareness of this issue, we have been working on the analysis of semantic structures in a news archive.

This paper first introduces two such attempts to analyze the semantics that lie in the contents of the news video archive; Section 2 introduces a work that analyzes the topic thread structures in the archive and also interfaces provided for the users to track up and down the structure, and Sect. 3 introduces a work that links news stories broadcast in different countries in different languages discussing the same event using both image and text information. After these works, a key technology required to assist the analysis in the image domain; near duplicate video segment detection, is introduced in Sect. 4. In the end, Sect. 5 concludes the paper. Details on each work should be referred to publications cited in each section.

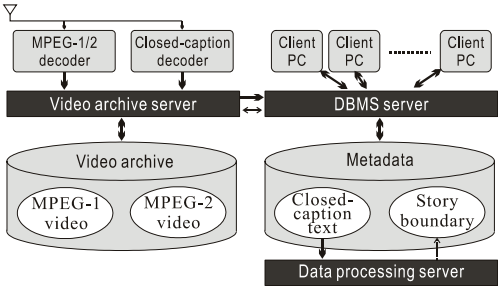

Figure 1: Broadcast news video archiving system.

Table 1: Specification of the news video archive.

News show	NHK "News7" (in Japanese)
Length	900− [hours] (20−30 [minutes/day])
Period	Mar. 16, 2001 (1,900− [days])
Data (Volume)	Video: MPEG-1/2, NTSC (550 [GB]/ 3.4 [TB]), Closed-Caption text (42 [MB])

2 Understanding news by the topic thread structure

In order to efficiently and effectively retrieve news footages based on their contents, it is necessary to analyze the semantic structure of the entire archive (Ide et al. 2006, Wu et al. 2006). In this section, we first describe how the news inherent structures are automatically analyzed in 2.1 (story segmentation). In 2.2, we then outline how the obtained structures are further analyzed; relations between the segmented stories are analyzed according to their chronological and semantic relations (topic threading). A detailed description of the threading method is provided in (Ide et al. 2006). In the end of this section, we will also introduce a topic tracking interface based on the thread structure. Topic tracking has been a strong interest in the text retrieval field, as seen in the TDT workshop series (US NIST). They define the term 'topic' as "*a seminal event or activity, along with all directly related events and activities*". Compared to this definition, the *topic thread* is slightly different in the sense that it connects gradually developing stories even across topics, where topic tracking generally terminates when it encounters a certain degree of gradual transition from the original story.

2.1 Story segmentation

To establish the topic thread structure, it is first necessary to extract the stories within a news show. The following

* Currently at DENSO Corp.

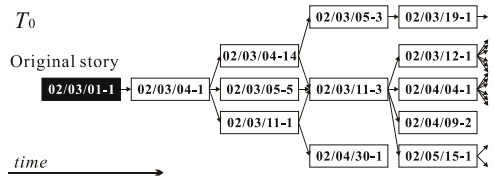

Figure 2: Example of a topic thread structure. Topics are labeled as [Year/Month/Day-Topic Number].

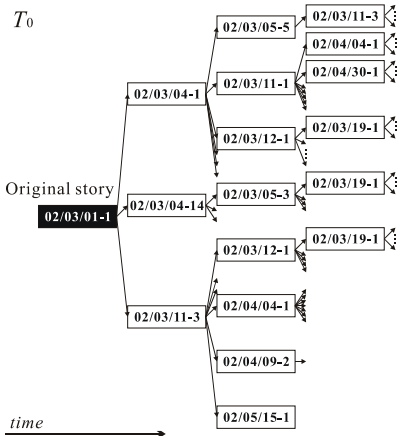

Figure 3: Example of a simple hierarchical story relation tree without threading.

process is applied to each sentence of a closed-caption text synchronized to the audio track:

1. Apply morphological analysis [1] to each sentence. Next, extract noun compounds according to the morphemes, followed by semantic attribute analysis by a suffix-based method (Ide et al. 1999).
2. Create keyword vectors for each sentence. Keyword vectors for four semantic attributes; general, personal, locational/organizational, and temporal, are formed by noun compounds extracted in Step 1.
3. At each sentence boundary, concatenate w (= 1 to 10 in the following experiments) adjacent vectors on both sides. Measure the similarity of the two concatenated vectors by the cosine measure, and choose the maximum similarity among all window sizes.
4. Sum up the similarities in each semantic attribute and detect a story boundary when it is smaller than θ_{seg}. According to a training with 384 manually given story boundaries, a weight of (general, personal, locational/ organizational, temporal) = (0.23, 0.21, 0.48, 0.08) for the summation and θ_{seg} = 0.17 were obtained.

An experiment applied to 130 manually annotated story boundaries as ground-truth, showed a precision of 90.5% and a recall of 95.4% if mis-judgments at a maximum of +/− 1 sentences were allowed.

2.2 Topic threading

Having segmented the various stories within a news show, we now describe how we establish the relations between them. The *topic thread* structure is a directed graph that connects related stories maintaining chronological orders

[1] JUMAN 3.61 distributed from Kyoto University was used.

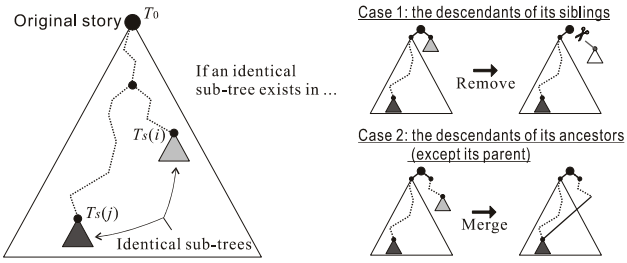

Figure 4: The topic threading scheme.

at each edge. The difference between the topic thread structure with a hierarchical tree that simply expands related stories at each node is that it lets a story appear only once in the tree where it is a child of the chronologically closest story (Compare Figs. 2 and 3). The topic thread structure is extracted as follows:

1. Expand a story relation tree recursively from the original story satisfying the following conditions:
   (a) Child nodes are stories related to their parent node, while at the same time, their time stamps succeed their parent's.
   (b) Siblings are sorted so that their time stamps succeed their left-siblings'.

   The relation between two stories is defined as the cosine measure between the keyword vectors of the stories. When its value exceeds a threshold θ_{trk}, the stories are considered related. This procedure forms a hierarchical story relation tree T_0 as shown in Fig. 3.
2. For each sub-tree $T_s(i)$ in the story relation tree T_0, if an identical sub-tree $T_s(j)$ exists on the left-side (in the future), apply either of the following operations:
   (a) Remove $T_s(i)$ if $T_s(j)$ is a descendant of $T_s(i)$'s sibling.
   (b) Else, merge $T_s(i)$ with $T_s(j)$ if $T_s(j)$ is a descendant of $T_s(i)$'s ancestor excluding its parent.

   The sub-tree is removed in (a) instead of merging, to avoid creating a shortcut without a story en route. The scheme is illustrated in Fig. 4. As a result, the thread structure forms a chronologically-ordered directed graph as shown in Fig. 2.

2.3 Topic clustering in the thread structure

In order to understand the semantic structure within the thread structure, topic clusters composed of similar stories along the thread are detected as follows:

1. Set S_O as both the cluster center ($N_0 = S_O$) and the current node ($N = S_O$).
2. Let child nodes of N be $N_c(j)$ (j = 1, ..., C). If none of the relations between N_0 and $N_c(j)$ exceed a threshold θ_{cls}, set N as the new cluster center ($N_0 = N$).
3. Apply Step 2. recursively until reaching all the leaf nodes $S_D(i)$ (i = 1, ..., L).

An example of the clustered thread structure is shown in Fig. 5.

2.4 Topic tracking interface: threadViewer

Figure 6 shows an interface that users may track up and down a thread structure originating from a specified news story, while browsing video segments corresponding to

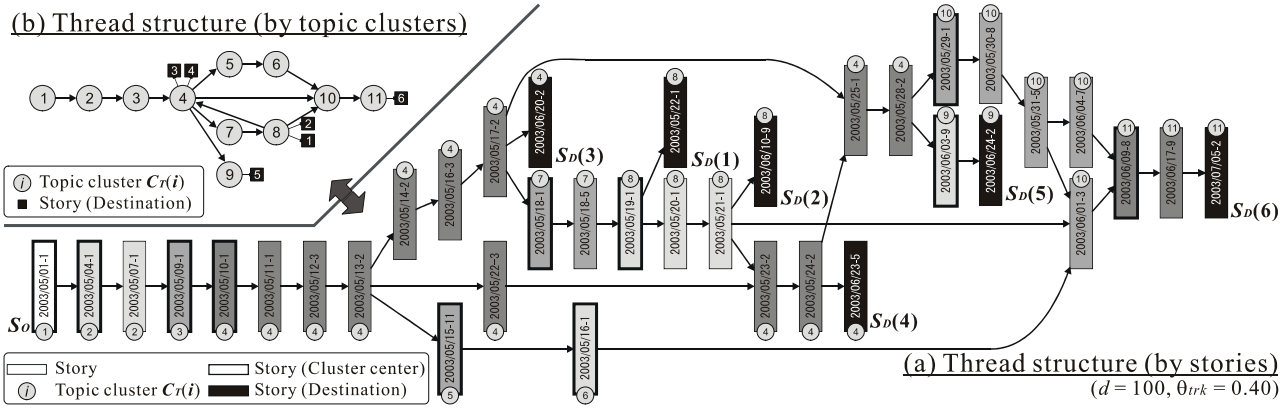

Figure 5: Topic thread structure originating from Story #1 on May 1, 2003. Video story V_S (S_O, S_D (6)) is composed as follows: Story S_O (= Topic cluster C_T (1)): SARS outbreak in Beijing; C_T (2): Epidemic spreads in mainland China; C_T (3): WHO sends a mission to Beijing; C_T (4): Epidemic slows down in mainland China, spreads in Taiwan; C_T (10): Epidemic calms down in mainland China, some infections reported in Toronto; C_T (11): Epidemic calms down in Taiwan; Story S_D (6): WHO announces the cease of the epidemic. It took 23 secs. to obtain this structure, including the threading and the clustering.

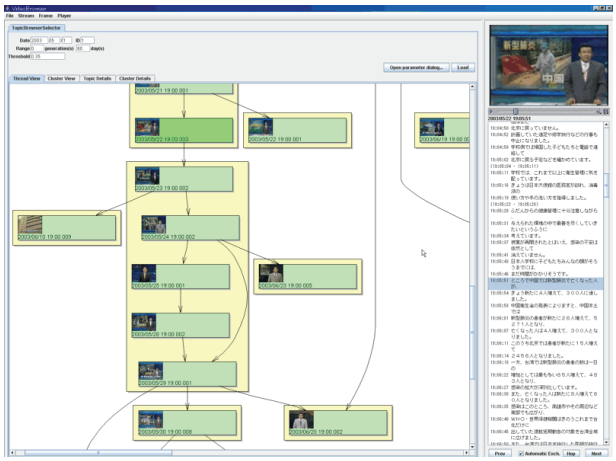

Figure 6: The "threadViewer" interface. The left side shows the thread structure, and the right side shows the actual video segment and the closed-caption text corresponding to a specified news story.

the stories. Such an interface allows the users to understand their topics of interest in detail.

3 Cross-lingual news story retrieval

Cross-lingual text retrieval has been a major research topic in the information retrieval field, especially by the TREC (US NIST a) community. However, conventional works have attempted to solve the problem solely in the text domain, which is generally considered more difficult than retrieval of texts in the same language.

Considering that visual information is language independent, in this section, we propose a cross-lingual news story retrieval method that exploits the existence of near-duplicate video segments in English and Japanese news shows, along with the similarity in the text domain.

Since the work is still in an early stage, we will simply analyze the results obtained from a preliminary experiment.

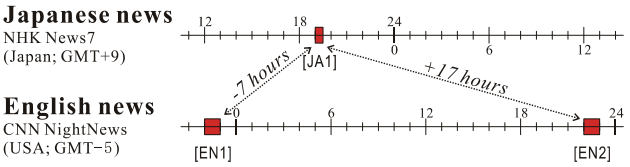

Figure 7: Time frame for comparing news shows in different languages for detecting stories discussing the same events.

3.1 Preparation

Story segmentation is necessary before comparing the similarity of stories. For the Japanese news show, the method introduced in Sect. 2.1 was applied to the closed-caption texts, but for the English news show, we manually segmented the transcripts obtained from the broadcaster's web page [2]. After obtaining the stories, the English texts were translated into Japanese by a machine translation software [3].

3.2 Comparison of news shows

The purpose of the method is to retrieve news stories that discuss the same events. Taking this in consideration, we assumed that such news stories should appear in a short time frame. Thus, as shown in Fig. 7, when a story-of-interest is set in a news show, corresponding stories are searched in news shows in other languages that were broadcast within +/− 24 hours.

In order to analyze the relation between two given stories, both the number of near-duplicate video segments and the similarity of keyword vectors are measured.

For the near-duplicate video segment detection, we employed the method described in Sect. 4. However, news videos obtained from different sources usually have a major difference in the frame layout, and also super-imposed captions are inserted in different locations

[2] http://transcripts.cnn.com/TRANSCRIPTS/
[3] "The Translation Professional" ver. 10 marketed by Toshiba Solutions Corp. was used.

Figure 8: Example of closely related news stories with near-duplicate segments but with few common keywords in the text. Two near-duplicate segments were detected, while only one keyword matched between the two stories. Such a case shows the effectiveness of employing the proposed method.

and timings. Thus, we only compare a fixed region in the vertical center of the frame which is usually not affected by such editing.

For the text comparison, noun compounds and undefined terms (usually proper nouns) are extracted by morphological analysis [4], and the cosine measure between keyword vectors is used to evaluate the similarity, as it was done in 2.2.

In the end, relations obtained from both image and text domains are combined. At this moment, we are still considering how to combine the relations obtained from the two domains.

3.3 Experiment

As a preliminary experiment, news shows listed in Tab. 2 were compared according to the temporal constraints shown in Fig. 7.

As a result, 26 pairs of near-duplicate video segments were found, which covered 4 pairs of news stories discussing the same event, and 1 pair which was actually not a near-duplicate segment. When the stories were compared by text, 4 pairs of news stories discussing the same event were found.

We then compared the pairs, and found that 2 pairs were found by both image and text clues, but 2 pairs were only found by the image clue, which shows the effectiveness of employing the proposed method together with the traditional text-based approach.

Figure 8 shows an example of closely related stories detected by the image clue and not by the text clue.

Table 2: News shows compared in the experiment.

Japanese news [5]	NHK "News7" (Japan: GMT+9) [JA1] Nov. 9, 2004; 7:00 pm − 7:30 pm
English news [6]	CNN "NewsNight" (USA: GMT−5) [EN1] Nov. 8, 2004; 10:00 pm − 11:00 pm [EN2] Nov. 9, 2004; 10:00 pm − 11:00 pm

[4] JUMAN 5.10 distributed from Kyoto University was used.

4 Efficient detection of near-duplicate video segments

This section introduces a general framework that detects all pairs of near-duplicate video segments efficiently from a long video stream (Length: n frames). Near-duplicate video segments are mostly identical video segments from

the image perspective, except for minor local differences such as overlay of captions or logos, or minor overall color difference. They are very important clues to understand semantic structures in video streams. Notice that it is different with the traditional similar video segment detection which detects not only mostly identical but also somewhat similar, but originally different video segments.

Detecting all pairs of near-duplicates in a video stream is, however, extremely time-consuming compared to detecting near-duplicates of a given segment, since it essentially requires computation of a square order of the video length ($O(n^2)$).

Conventional methods approach this problem by accelerating the detection through a two step detection process (Sekimoto et al. 2000, Naturel & Gros 2005, Yamagishi et al. 2003, Yang et al. 2005). The common idea of the methods, including our method, is to make the $O(n^2)$ times comparison as fast and accurate as possible by applying a fast and rough search first and then checking accurately afterwards. These methods do accelerate the detection, but they do not necessarily guarantee that the detection in the first step has no false negatives.

Considering this problem, we propose a method that guarantees that it has no false negatives in the first step, where the results would be theoretically equivalent to the brute-force frame-by-frame comparison. The basic idea of the method is to make the $O(n^2)$ times of comparison as fast and accurate as possible by 1) comparing the features of short video fragments instead of a frame, and also by 2) compressing the dimension of the feature vectors, as shown in Fig. 9. The first approach makes the comparison

[5] Source: NII news video archive
[6] Source: TRECVid 2004 data (US NIST b)

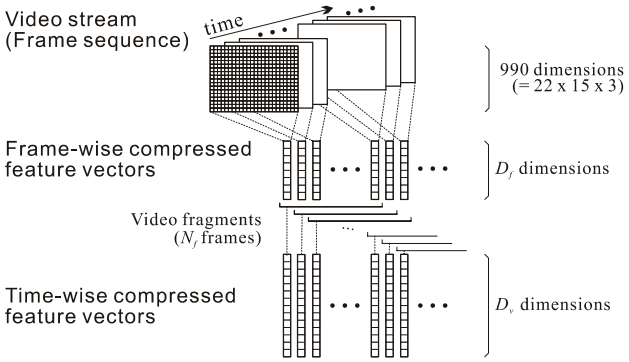

Figure 9: Spatiotemporal feature vector compression.

efficient and at the same time robust to noise. Meanwhile, the second approach reduces computation time together with i/o time and storage space which is a significant problem when processing a long video stream.

4.1 Selecting video features for compression

In order to compress video features efficiently, it is necessary to select features more informative than others. In the proposed method, PCA (principal component analysis) is applied to feature vectors composed of raw pixel values obtained from each frame of a training video stream.

For preparation, each frame i extracted from an MPEG-1 video stream (Original size: 352 x 240 pixels) is first degraded to 22 x 15 pixels. Next, a feature vector $\boldsymbol{f}_i = {}^t[f_{i,1}, f_{i,2}, \ldots, f_{i,m_f}]$ is composed by arranging the RGB values $f_{i,j}$ of all the pixels as an array, which forms an $m_f = 990$ (= 22 x 15 x 3) dimension vector. In order to absorb color difference among different video sources, $\boldsymbol{f}_i$ is normalized to $\hat{\boldsymbol{f}}_i$ (hereafter, *frame vector*) as follows:

$$\mu_i = \sum_{j=1}^{m_f} f_{i,j} \Big/ m_f \tag{1}$$

$$\bar{\boldsymbol{f}}_i = {}^t\left[f_{i,1} - \mu_i, f_{i,2} - \mu_i, \ldots, f_{i,m_f} - \mu_i\right] \tag{2}$$

$$\hat{\boldsymbol{f}}_i = \bar{\boldsymbol{f}}_i \Big/ \left\|\bar{\boldsymbol{f}}_i\right\| \tag{3}$$

As shown in Fig. 9, the compression is realized by selecting features by the following two steps:

1. Frame-wise feature selection

   Create a matrix $\boldsymbol{F} = [\,\hat{\boldsymbol{f}}_{i,1}\hat{\boldsymbol{f}}_{i,2}\ldots\hat{\boldsymbol{f}}_{i,K_f}\,]$ by arranging $K_f\,(\geq m_f)$ frame vectors. Obtain the unit eigenvectors $\{\,\boldsymbol{e}_{f,1}, \boldsymbol{e}_{f,2}, \ldots, \boldsymbol{e}_{f,D_f}\}$ corresponding to the top $D_f\,(\leq m_f)$ large eigenvalues of the auto-correlation matrix $\boldsymbol{Q}_{\boldsymbol{F}} = \boldsymbol{F}\,{}^t\boldsymbol{F}$. The vector space spanned by the basis $\langle\ \boldsymbol{e}_{f,1}, \boldsymbol{e}_{f,2}, \ldots, \boldsymbol{e}_{f,D_f}\ \rangle$ is used as the frame-wise compressed feature space.
2. Time-wise feature selection

   For each frame, transform the frame vector $\hat{\boldsymbol{f}}_i$ on to the vector space $\langle\ \boldsymbol{e}_{f,1}, \boldsymbol{e}_{f,2}, \ldots, \boldsymbol{e}_{f,D_f}\ \rangle$ to obtain a frame-wise compressed vector $\boldsymbol{f}'_i$ as follows:

$$\boldsymbol{f}'_i = {}^t\left[\boldsymbol{e}_{f,1}\boldsymbol{e}_{f,2}\ldots\boldsymbol{e}_{f,D_f}\right]\hat{\boldsymbol{f}}_i \tag{4}$$

   Next, N_f adjoining compressed frame vectors starting from frame i are concatenated as a spatiotemporal feature vector $\hat{\boldsymbol{v}}_i$ with a dimension of $m_v = D_f N_f$:

$$\hat{\boldsymbol{v}}_i = {}^t\left[{}^t\boldsymbol{f}'_i, {}^t\boldsymbol{f}'_{i+1}, \ldots, {}^t\boldsymbol{f}'_{i+N_f-1}\right] \tag{5}$$

   where N_f is the size of the video fragments. A matrix $\boldsymbol{V} = [\,\hat{\boldsymbol{v}}_{i,1}\hat{\boldsymbol{v}}_{i,2}\ldots\hat{\boldsymbol{v}}_{i,K_v}\,]$ is created by arranging the K_v $(\geq m_v)$ spatiotemporal vectors. Again, obtain the unit eigenvectors $\{\,\boldsymbol{e}_{v,1}, \boldsymbol{e}_{v,2}, \ldots, \boldsymbol{e}_{v,D_v}\}$ corresponding to the top D_v $(\leq m_v)$ large eigenvalues of the auto-correlation matrix $\boldsymbol{Q}_{\boldsymbol{V}} = \boldsymbol{V}\,{}^t\boldsymbol{V}$. The vector space spanned by the basis $\langle\ \boldsymbol{e}_{v,1}, \boldsymbol{e}_{v,2}, \ldots, \boldsymbol{e}_{v,D_v}\ \rangle$ is used as the time-wise compressed feature space.

Thus is obtained the compressed feature space that efficiently represent the video segments.

4.2 Detecting near-duplicate video segments

4.2.1 Extraction of spatiotemporal feature vectors

The same preparation as in the training phase described in 4.1 is applied to an input video stream. Likewise, a normalized input frame vector $\hat{\boldsymbol{f}}_i$ is frame-wise compressed by Eq. 4. It is then concatenated with the following $N_f - 1$ compressed frame vectors as in Eq. 5 ($\hat{\boldsymbol{v}}_i$) and then time-wise compressed by the following transformation:

$$\boldsymbol{v}'_i = {}^t\left[\boldsymbol{e}_{v,1}\boldsymbol{e}_{v,2}\ldots\boldsymbol{e}_{v,D_v}\right]\hat{\boldsymbol{v}}_i \tag{6}$$

In this manner, compressed spatiotemporal feature vectors are created for all video fragments by shifting a window with a size of N_f frames, frame by frame.

4.2.2 Step 1: Comparison of video fragments in the compressed feature space

Video fragments i_1, i_2 are compared by the L_2 distance d_1 between their spatiotemporal feature vectors as follows:

$$d_1(\boldsymbol{v}'_{i_1}, \boldsymbol{v}'_{i_2}) = \sqrt{\sum_{l=1}^{D_v}\left(v'_{i_1,l} - v'_{i_2,l}\right)^2} \tag{7}$$

where $v'_{i,l}$ represents the l-th component of a vector $\boldsymbol{v}'_i$. When d_1 is shorter then a threshold θ_1, the fragments are considered as near-duplicate candidates. Note that when $d_1 \leq \theta_1$, we can always expect that d_1 is always shorter than the distance in the original space, due to the fact that L_2 distance is always shorter in a subspace of an eigenspace than in the original space. This property guarantees that the detection in the compressed feature space has no false negatives of video fragments longer than N_f frames that should be detected in the original feature space.

Theoretically, it is necessary to compare all fragments versus all fragments, which results in ${}_{n-N_f+1}C_2$ times of comparison for a video stream with a length of n frames. This may however, be reduced if we can restrict the minimum length of a near-duplicate video segment that should be detected. If the minimum length is set to N_{min} frames long, it is possible to skip $N_h = N_f - N_{min}$ frames on one side of the comparison, which results in reducing the

Table 3: Parameters used in the experiment.

K_f, K_v	15,000	$\theta_1 = \theta_2$	0.8
D_f	10 [dimensions]	N_f	150 [frames]
D_v	10 [dimensions]	N_h	150 [frames]

total times of comparison. As a matter of fact, when we consider general broadcast video streams as a target, the combination of N_f and N_h could be set to relatively high numbers depending on the application. For example, when detecting all commercials longer than N_{min} = 900 frames (30 seconds) and possibly some of those longer than 450 frames (15 seconds), N_f = 450, N_h = 450 may be a good combination.

4.2.3 Step 2: Confirmation of near-duplicate fragments in the original feature space

The comparison of video fragments in the low-dimension feature space derives numerous candidates of near-duplicate video fragment pairs. In order to filter out false positives among the candidates, the pairs are compared in the original (high-dimension) feature space by the following function:

$$d_2(\boldsymbol{v}_{i_1}, \boldsymbol{v}_{i_2}) = \sqrt{\sum_{l=1}^{m_v} \left(v_{i_1,l} - v_{i_2,l}\right)^2} \qquad (8)$$

where $\boldsymbol{v}_i = {}^t[\,{}^t\boldsymbol{f}_i, {}^t\boldsymbol{f}_{i+1}, \ldots, {}^t\boldsymbol{f}_{i+Nf-1}] = {}^t[v_{i,1}, v_{i,2}, \ldots, v_{i,mv}]$ and $m_v = m_f N_f$. When $d_2 \le \theta_2$, the fragments are confirmed as near-duplicates.

4.2.4 Post-processing

After detecting near-duplicate fragments, precise boundaries of near-duplicate segments are obtained as a final result by adjusting the boundaries by frame-by-frame comparison at both ends of the fragments.

4.3 Experiment

The method was applied to broadcast video data to count the computation time on a Pentium 4 3.0GHz PC with 1.0GB of main memory. Parameters were set to the values shown in Tab. 3. As training data, 150 hours of continuous video stream obtained from a Japanese channel during June 1−7, 2004 were used. The sample frames/fragments were selected randomly.

The computation time by brute-force frame-by-frame comparison was also measured with the same similarity threshold, while its result was referred to as the ground-truth. The parameters ensure the detection of near-duplicate video segments at least 300 frames (10 seconds) long.

The computation time counted by 1, 3, 6, 12, and 24 hours of general video streams with many near-duplicates (mostly commercials) is shown in Fig. 10. The result shows that the proposed method significantly reduces the computation time required for the detection. Although the process was more than 1,000 times faster, there was not a single false positive nor a false negative against the brute-force frame-by-frame comparison, which we consider as the ground-truth. In that sense, the proposed method outputs results identical to the results longer than N_{min} frames from the brute-force frame-by-frame comparison, which makes the precision and the recall both

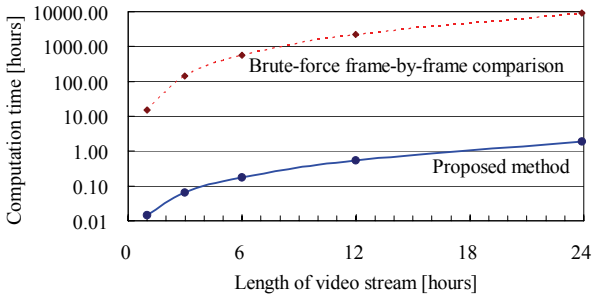

Figure 10: Computation time for the detection.

100%, while reducing the computation time drastically. From the result, we estimate that it should be possible to reduce the computation time required for a 1 week long video stream from 50 years to 5 days.

5 Conclusion

In this paper, we introduced our current research activities on handling the contents of a large-scale news video archive. In the future, we will keep working on developing basic technologies scalable enough to handle larger volumes of video data, and at the same time, use the technologies effectively in applications and interfaces for better understanding of voluminous video data.

Acknowledgements

We would like to thank our colleagues and former/current students, especially Dr. Norio Katayama and Dr. Hiroshi Mo at the National Institute of Informatics, together with Dr. Tomokazu Takahashi, at Nagoya University.

Parts of the works were funded by the Grants-in-Aid for Scientific Research (15700116, 16016289, 18049035, 18700080) and the 21st Century COE program from the Ministry of Education, Culture, Sports, Science and Technology, and also by the Research Grant (K17ReX-202) from Kayamori Foundation of Information Science Advancement.

References

Ide, I., Hamada, R., Sakai, S. & Tanaka, H. (1999), Semantic analysis of television news captions referring to suffixes, *in* 'Proc 4th Intl. Workshop on Information Retrieval with Asian Languages', pp.37−42.

Ide, I., Mo, H., Katayama, N. & Satoh, S. (2006), Exploiting topic thread structures in a news video archive for the semi-automatic generation of video summaries, *in* 'Proc IEEE 2006 Intl. Conf. on Multimedia and Expo', pp.1473−1476.

Naturel, X. & Gros, P. (2005), A fast shot matching strategy for detecting duplicate sequences in a television stream, *in* 'Proc. 2nd Intl. Workshop on Computer Vision meets Databases', pp.21−27.

Sekimoto, N., Nishimura, T., Takahashi, H. & Oka, R. (2000), Continuous retrieval of video using segmentation-free query, *in* 'Proc. 15th Intl. Conf. on Pattern Recognition', pp.375−378.

U.S. National Institute of Standard and Technology (a), 'Text REtreival Conf. (TREC)', http://trec.nist.gov/.

U.S. National Institute of Standard and Technology (b), 'TRECVid evaluation', http://www-nlpir.nist.gov/projects/trecvid/.

U.S. National Institute of Standards and Technology, 'Topic detection and tracking (TDT)', http://www.nist.gov/speech/tests/tdt/.

Wu, X., Ngo, C.-W. & Li, Q. (2006), 'Threading and autodocumenting news videos', *IEEE Signal Processing Magazine* **23**(2), 59−68.

Yamagishi, F., Satoh, S., Hamada, T. & Sakauchi, M. (2003), Identical video segment detection for large-scale broadcast video archives, *in* 'Proc. 3rd Intl. Workshop on Content-Based Multimedia Indexing', pp.135−141.

Yang, X., Xue, P. & Tian, Q. (2005), A repeated video clip identification system, *in* 'Proc. 13th ACM Intl. Conf. on Multimedia', pp.227−228.