Unsupervised Video Semantic Partitioning Using IBM Watson And Topic Modelling

Alexandre Quemy
IBM Software Lab
Cracow, Poland
Faculty of Computing, Poznań University of Technology
Poznań, Poland
aquemy@pl.ibm.com

Krzysztof Jamrog
IBM Software Lab
Cracow, Poland
krzysztof.jamrog1@pl.ibm.com

Marcin Janiszewski
IBM Software Lab
Cracow, Poland
marcin.janiszewski@pl.ibm.com

ABSTRACT

In this paper, we present the problem of Video Semantic Partitioning that consists in breaking a video into semantically independent blocks. Defined within a framework of optimization, we present a preliminary heuristic approach to solve the problem, called Split-and-Merge. The algorithm itself is unsupervised, but the mechanisms to extract data from videos are supervised (for some) since they used IBM Watson Services. Finally, we demonstrate on few videos the capabilities of our prototype and discuss the limitations and future improvements. From the experiments, we draw two conclusions: (i) the optimal solution to the problem varies from human to human with a large variability from video to video, (ii) Split-and-Merge demonstrates encouraging qualitative results to find the *average* optimal solution defined as the average solution given by humans.

1 INTRODUCTION

The problem of text segmentation, that is to say partitioning a text into semantic blocks, has been widely studied (e.g. in [2, 5]) but, as far as we know, never extended to videos. Indeed, video segmentation usually refers to the process of extracting objects from a video and not breaking it into semantic chunks. In this paper, we define the Video Semantic Partitioning problem as an extension of the text segmentation problem and present our primary attempt to solve it. Among the possible applications, we can highlight a better understanding of the video content for tagging and indexing in a database or the creation of educational paths by joining semantic blocks from different videos to create a new one.

A video is a complex and multidimensional signal holding a lot of information in a very short amount of time. In addition, the *raw* information is totally unstructured and has to be extracted, contrary to text segmentation where this information is mostly structured (the text is already provided as it is). By *raw*, we mean the primary source of semantic. For a given text, it mainly holds in the sentences and words themselves, while the support brings some secondary elements (e.g. in [2] the support is a HTML page and the tags are used to delimit some sections). For a video, the primary source is not as well defined: is it the audio track or the visual information? Most likely a combination of both. Additionally, this raw information is not directly suitable for most algorithms and requires a pre-treatment to extract exploitable data. Those techniques are known as *Cognitive Computing*, i.e. processing and analyzing large unstructured signals. The information extraction plays a preponderant role in the algorithms performances because even the best possible algorithm would return poor results on badly extracted information which is equivalent to noise.

The plan of this paper is as follows: Section 2 formalizes the Video Semantic Partitioning problem, Section 3 presents the feature extraction mechanisms while in Section 4 the algorithm Split-and-Merge is presented. In Section 5 we discuss an optimized version of the algorithm under a restrictive assumption about the videos. In Section 6, we present some results obtained on real videos and conclude this paper in Section 7.

2 VIDEO SEMANTIC PARTITIONING

A video is a temporal signal on a time interval $[0, T]$. We denote by $P([0,T], m)$ the set of partitions of $[0, T]$ into m elements, i.e. the elements of $P([0,T], m)$ are of the form $\pi = (p_0, ..., p_m, p_{m+1})$, $\forall i \in \{0, ..., m\},\ p_i < p_{i+1}$, with $p_0 = 0$ and $p_{m+1} = T$, such that $\bigcup_{i=0}^{m} [p_i, p_{i+1}] = [0, T]$.

Our goal is to find a partition such that each $[p_{i-1}, p_i]$ is an independent semantic block, i.e. holds a different semantic than its adjacent blocks. Let π^* denotes the optimal partition. The first formulation of the Semantic Partition problem is to find a minimizing sequence π_k such that $\pi_k \underset{k}{\to} \pi^*$ with a convergence notion that implies that the number of cutting points of π_k converges to those of π^*[1] and that each cutting point of π^* is the limit of a cutting point of π_k.

As an element of any partition π is contained in a closed interval defined by two cutting points of π^* we can define the following cost function $\forall \pi \in \bigcup_{m \in \mathbb{N}^*} P([0,T], m)$, $\forall p_i \in \pi,\ c(p_i) = |p_j^* - p_i||p_i - p_{j+1}^*|$ with $p_i \in [p_j^*, p_{j+1}^*]$. Additionally, we define $J(\pi, \pi^*) = \sum_{i=1}^{m-1} c(p_i) + c(p_{i+1})$.

Under the assumption that π has as many cutting points as π^*, then if $J(p, p^*) = 0$ then $p = p^*$ and the problem can be seen as finding a minimizing sequence π_k s.t. $J(\pi_k, \pi^*) \underset{k}{\to} 0$. In practice, as π^* is unknown, there are two problems: the optimal number of cutting points is not known (i) and J is not available (ii).

For (i), assuming J is available, if π_0 contains enough cutting points and some in each interval induced by π^*, we can minimize J given π_0 by removing the unnecessary cuttings points. Assuming we know how to remove the unnecessary cutting points, if $\pi_{0,h}$ is defined by the uniform partition s.t. $\forall p_i, p_{i+1} \in \pi_0,\ p_{i+1} - p_i = h$, then $\forall h_1, h_2$ s.t. $h_1 < h_2$, $J(\pi_{0,h_1}, \pi^*) < J(\pi_{0,h_2}, \pi^*)$. More generally, we can control J by h.

[1] We may not accept the case such that two cutting points are identical, creating an empty interval, thus with an empty semantic. However, it may be convenient for practical algorithms and thus we assume we merge the similar cutting points of π_k at any time if necessary.

For (ii), we need to find an approximation of J. As p^* is a partition into *semantic* blocks we will try to rely on the semantic contained in the video to approximate J. To do so, we are able to identify n features *stream* from the video, that is to say, n sequences of features on $[0,T]$. For each $i \in \{1,...,n\}$, we have a similarity metric $d^i : [0,T]\times[0,T] \to [0,1]$ to quantify how related two blocks are w.r.t. a particular feature *stream*. In particular, $\forall p_1,p_2,\ d_i([p_1,p_2],[p_1,p_2]) = 1$ because the semantic of a block is obviously equal to its own semantic. However, $[p_1,p_2]\cap[p_3,p_4] = \emptyset$ does not imply $d_i([p_1,p_2],[p_3,p_4]) = 0$, i.e. it is not because two blocks are not overlapping in time that they do not have semantic similarity. To ease the notations, given a partition π, we denote by d_i^j the similarity between the blocks $[p_{j-1},p_j]$ and $[p_j,p_{j+1}]$, i.e. $d_i^j \overset{def}{=} d_i([p_{j-1},p_j],[p_j,p_{j+1}])$.

The problem of the video partitioning is to find a partition $\pi \in \bigcup_{k\in\mathbb{N}^*} P([0,T],m)$ that minimizes the adjacent block similarity, that is to say

$$\pi^* = \underset{\pi\in\bigcup_{m\in\mathbb{N}^*}P([0,T],m)}{\operatorname{argmin}} \left[\bigoplus_{1\le i\le n}\sum_{j=1}^{m} d_i^j\right] \tag{1}$$

with $\bigoplus$ an aggregation operator.

For the purpose of this paper, we will consider $\bigoplus$ as a convex combination of the block similarity, allowing us to reformulate the problem as follows:

$$\begin{cases} \pi^* = \underset{\pi\in\bigcup_{m\in\mathbb{N}^*}P([0,T],m)}{\operatorname{argmin}} \left[\sum_{1\le i\le n}\sum_{j=1}^{m}\omega_i\, d_i^j\right] \\ \sum_{1\le i\le n}\omega_i = 1 \end{cases} \tag{2}$$

The choice of the operator $\bigoplus$ is an interesting problem by itself because it is known to greatly influence the algorithms capacity to find the optimal solution [7]. In the future, we plan to investigate the choice of this operator, and in particular treating the problem as multi-objective (optimizing on each feature stream independently) using the Hypervolume [1] or ε-dominance [4] metric as aggregation function.

As said before, one major difficulty comes from the fact the optimal number of elements in the partition is not known *a priori*. Notice that in practice, if an interval is too small the information it holds is considered as null. For instance, a very small interval does not hold more than few words or no word at all, no screenshot, etc. As a result, its similarity with another very small interval would be one or close to one. In other words, dividing too many times $[0,T]$ does not minimize the objective function. However, having two large blocks separated by a very small block would be a minimizing strategy: the small block holds no semantic, contrary to the large blocks and as a result, the similarity between adjacent blocks is (close to) zero. To handle this problem, we see several possibilities: taking into account this phenomenon into the definition of the metrics d_i (i), adding a regularization term on the block sizes to (2) to be sure they are homogenous enough (ii), fixing the set of possible values for m and a distribution of block size w.r.t. T (iii). As (i) is too ad-hoc and we do not see any practical justification to support (ii), we will make a stronger assumption on the optimal partition in Section 5 that will define both the maximal number of blocks and their respective sizes.

3 FEATURE STREAMS

In this section, we review the feature streams we extract from a video as well as the associated block similarity.

We are able to extract a frame at any moment of the video. For two frames, we are able to calculate two metrics: a perceptual hash $d_{\texttt{hash}}$ using `phash` open-source library[2] and a pixel-based hamming distance $d_{\texttt{pixel}}$. For two blocks $[p_{j-1},p_j]$ and $[p_j,p_{j+1}]$, we made the choice to take into account resp. only the last and the first frame contain in the first and second block. With Watson Visual Recognition and for each frame, we are able to obtain a list of tags with confidence on concepts and entities. For a block $[p_j,p_{j+1}]$, we aggregate the tags of all frames it contains into a the sets W_j and C_j respectively for the words and the confidence level. For two blocks, we denote by $W_{j,j+1}$ the set defined by $W_j \cap W_{j+1}$ with $C^L_{j,j+1}$ and $C^R_{j,j+1}$ the confidence associated to the left and right block.

Example: Let us assume $W_j = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$ composed of three words, associated to the confidence vector $C_j = \begin{pmatrix} c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix}$. Similarly, let us assume $W_{j+1} = \begin{pmatrix} w_1 \\ w_3 \\ w_4 \end{pmatrix}$ with $C_{j+1} = \begin{pmatrix} c_{1,j+1} \\ c_{3,j+1} \\ c_{4,j+1} \end{pmatrix}$. Then, $W_{j,j+1} = \begin{pmatrix} w_1 \\ w_3 \end{pmatrix}$ and the left and right confidence vectors are given by $C^L_{j,j+1} = \begin{pmatrix} c_{1,j} \\ c_{3,j} \end{pmatrix}$ and $C^R_{j,j+1} = \begin{pmatrix} c_{1,j+1} \\ c_{3,j+1} \end{pmatrix}$.

If $|W_{j-1}\cap W_j| = 0$, the similarity is zero: $d^j_{\texttt{tags}} = 0$. Otherwise, the similarity measure is defined as the Jaccard index ponderated by the confidence:

$$\begin{aligned} d^j_{\texttt{tags}} &= \frac{|W_{j-1}\cap W_j|}{|W_{j-1}\cup W_j|}\left(1 - \frac{||C^L_{j-1,j} - C^R_{j-1,j}||_1}{|W_{j-1}\cap W_j|}\right) \\ &= \frac{|W_{j-1}\cap W_j| - ||C^L_{j-1,j} - C^R_{j-1,j}||_1}{|W_{j-1}\cup W_j|} \end{aligned}$$

where $|A|$ denotes the cardinal of A. In other words, two blocks are similar not only if they have the same tags but also if their confidences are close enough.

Example: Considering the vectors defined in the previous example, the similarity between the two blocks is

$$d^{j+1}_{\texttt{tags}} = \frac{2 - [(c_{1,j} - c_{1,j+1}) + (c_{3,j} - c_{3,j+1})]}{4}$$

In this preliminary work, we did not consider OCR to extract the text from the frames or image segmentation techniques to obtain the location of objects on a frame. However, using Watson Speech-to-Text, we extract the transcript from the audio track. The service provides the timestamp to be able to locate the text corresponding to a given block. Similarly to Watson Visual Recognition, Watson Natural Language Understanding is able to extract from a text a list of keywords and concepts with a confidence level (expressing the *relevance* among the text). This allows us to define $d^j_{\texttt{keywords}}$ and $d^j_{\texttt{concepts}}$ in a similar way as for $d^j_{\texttt{tags}}$.

We normalized the transcript using `NLTK` [6] which includes the tokenization, removing the stopwords, the lemmatization,

[2] http://phash.org/

the computation of n-grams from 1 to 4 and filtering the n-grams by a minimal number of occurrences. A given partition $\pi \in P([0,T],m)$ can be seen as a corpus of m documents represented by Bag-of-Words. Using `Gensim` [8], we applied a TF-IDF transformation on each block and used a Latent Dirichlet Allocation [3] to express each block in a latent topic space. LDA is a generative probabilistic model that represents documents as a mixture of topics. Each topic represents a (sparse) distribution over the dictionary of words used in the documents expressing the idea that a topic is defined by a high frequency of few terms. The model parameters are then estimated from the real data, in particular the word distribution per topic and the topic distribution for each document. For each pair of adjacent blocks, we use the LDA model to determine the distance between the blocks denoted by $d^j_{\texttt{topic}}$. LDA has three hyperparameters: K the number of topics, α the document-topic prior distribution and η the topic-word priori distribution. The last two parameters control the sparsity of the distribution. In this work, we used the online hyperparameter strategy provided by `Gensim`, i.e. the values for α and η are deduced from the real data. As LDA can be seen as a dimensionality reduction method, the number of dimensions K influences the quality of the model, thus should be fixed not too low to keep enough information but not too high for the documents to be summarized. For this paper, knowing the approximate size of the Bag-of-Words representation of our data, we fixed $K = 10$. Further work should focus on hyperparameter tuning taking into account the length of the video and the number of cutting points at a given moment in a partition (because it influences the transcript length of a block).

We would like to draw the attention on the fact that if the algorithm we used is not supervised, the feature extraction is supervised for some feature streams. The transcript obtained with Watson Speech-to-Text is rectified (including abbreviation and specific term annotations) and inserted into a custom model. As a result, from video to video, the feature extraction becomes more and more relevant, which we believe is the primary requirement to achieve a good semantic partitioning. Similarly, Watson Visual Recognition service is trained to recognize certain objects in relation with the theme of the test videos (e.g. trained to recognize specific softwares displayed in the video).

4 SPLIT-AND-MERGE ALGORITHM

The algorithm is composed of two phases. In the first one, called Split, we break the video into regular blocks that are small enough to be sure they are smaller than the smallest semantic block of the video and thus, in particular, there are several cutting points within each of the intervals defined by the optimal partition p^*. The Merge phase consists in merging two adjacent blocks if they are related enough according to the metrics we defined in the previous Section. Iteratively, we merge the blocks with the higher normalized combined distance defined by $F(\pi) = \frac{1}{n|\pi|}\sum_{1\le i\le n}\sum_{j=1}^{|\pi|}\omega_i\, d_i^j = \frac{1}{n|\pi|}\sum_{j=1}^{|\pi|}F^j(\pi)$ with $F^j(\pi) = \sum_{i=1}^{n}\omega_i d_i^j$. However, without a stopping criterion all the blocks will be merged. To prevent this, we take into consideration the improvement implied by removing p_j between step k and $k+1$: $R(\pi_k, j) = \frac{F(\pi_{k+1})}{F(\pi_k)}$ with $\pi_{k+1} = \pi_k \setminus \{p_j\}$. Split-and-Merge is detailed in Algorithm 1. It has four hyperparameters: $m > 0$, $\omega_i \in [0,1]$ s.t. $\sum_i \omega_i = 1$, $\delta \in [0,1]$ and $\eta \in [0,1[$, resp. the initial number of cutting points, the weights for each feature, a similarity threshold under which we do not want to merge two blocks and an improvement threshold under which we consider it is not interesting enough to merge.

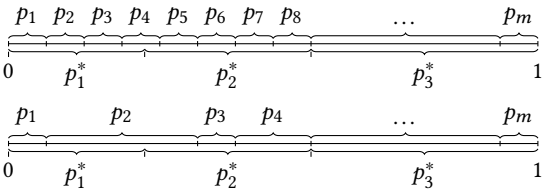

Figure 1: Illustration of Split-and-Merge. On top, the initial uniform partition, finer than the optimal partition π^*. At the bottom, the partition after few steps: several cutting points have been removed. It is to be noticed that, as $p_1{}^*$ or $p_2{}^*$ are not on the initial partition, it is impossible to have them in the output of Split-and-Merge.

Algorithm 1 Split-and-Merge

$\pi_0 \leftarrow \{\frac{T}{m}i\}_{i=0}^m$
do
  Calculate the family $\{d_i^j\}_{i,j}$ for π_k
  $F^j \leftarrow \frac{1}{n}\sum_{i=1}^{n}\omega_i d_i^j$
  for $F^j \in \{F^j \mid F^{j_1} \ge F^{j_2},\ \forall j_1 > j_2\}$ **do**
    if $F^j \ge \delta$ and $R(\pi_k, j) \le \eta$ **then**
      $\pi_{k+1} = \pi_k \setminus \{p_j\}$
      break
    end if
  end for
while $\pi_k \ne \pi_{k+1}$

5 RESTRICTING THE SEARCH SPACE

If the audio of the video is intended to describe the video content (e.g. a documentary or a commented video for educational purposes), we make a stronger assumption: a cutting point between two semantic blocks can only occur between two sentences. Not only this assumption turns the continuous problem into a (almost) discrete problem, but it also provides an upper bound on the number of cutting points. If we denote by $S = \{s_1, ..., s_K\}$ the set of sentences in the perfect transcript, the number of cutting points is bounded by K. It is only almost discrete, because in many cases the time interval between two sentences may be quite long and there is still a need to determine where to cut exactly.

Let $[\varepsilon^+(s_j), \varepsilon^-(s_{j+1})] \subset [0,T]$ denotes the time interval between two consecutive sentences s_j and s_{j+1}. In other words, $\forall p \in \pi^*,\ \exists j \in \{1, ..., K-1\}$ s.t. $p_j \in [\varepsilon^+(s_j), \varepsilon^-(s_{j+1})]$. To take advantage of this assumption, we change the initial uniform partition π_0 defined in Algorithm 1 By $\pi_0 = \{p_i \mid \forall i \in \{1, ..., K-1\},\ p_i = \varepsilon^+(s_i) + \frac{|\varepsilon^-(s_{i+1}) - \varepsilon^+(s_i)|}{2}\}$. Contrary to the previous version, there is no parameter h to control the (best case) precision. However, as we know there is a unique cutting point in $[\varepsilon^+(s_j), \varepsilon^-(s_{j+1})]$, we can split it into an arbitrary number of potential cutting points and select the one that minimizes the aggregated similarity. This step is done after applying Split-and-Merge as a refinement step.

6 EXPERIMENTS AND RESULTS

The plan of this section is as follows: we present the data we used in Section 6.1, then we elaborate the protocol in Section 6.2. We analyse the results in Section 6.3. Last but not least, in Section 6.4, we address some problems or possible critics of our protocol.

6.1 Videos

We used three short videos (2:16 to 4:00 minutes) available on Youtube:

(1) Running an experiment in the IBM Quantum Experience[3]
(2) IBM Bluemix - 3 minutes demo (NodeJS, SSO)[4]
(3) IBM Watson for Oncology Demo[5]

Apart from being short, the videos have some specific characteristics. The first video presents how to create, run and access to the result of a quantum algorithm. The video is relatively naturally broken down into parts except the moment when it explains the algorithm where the video switches between displaying a screen with the software to perform the experiment and an illustration using cards. Also, it seems to us that there is a long block in the middle of the video which would require Split-and-Merge to really understand how to remove inappropriate cutting points. The second one is very naturally broken down into several parts by a short notice with the subject of the next part written during the transitions. Between transitions, the speaker is visually present on the video which can perturbate the visual metrics since he tends to move, thus modifying perceptual hash and pixel-based distance more than expected. Finally, the third video has been selected because it is harder to break down since the speaker flow and presentation seem more "continuous" without neat transition between screens or topics. Also, it integrated a visual transition (from black screen to a web page and the contrary) at the beginning and at the end, which can trick both humans and the visual metrics.

6.2 Protocol

We asked people through a public form to indicate their optimal partition with the following instruction: *For each video, describe how you would break it into "steps" (a bit like a chapter in a book).* We added that in case of doubt, one can indicate that a transition is *weak*. We respectively collected seven, five and five answers for each video[6]. For each answer we obtained, we plot the cutting points t_i as a three second interval centered on t_i (in blue for the weak cutting points, red otherwise) such that we can visually observe the areas on the timeline such that people agree there is a cutting point.

Example of answer for the first video:

```
0:12
0:20 W
0:31 W
0:38
0:59
2:35 W
2:56 W
3:49 W
```

[3]https://www.youtube.com/watch?v=pYD6bvKLI_c
[4]https://www.youtube.com/watch?v=xE_I8BgDroA
[5]https://www.youtube.com/watch?v=338CIHlVi7A
[6]Due to the required time to complete the form, people were able to answer only for some videos. On top of that, a fourth video was initially planned, but as we received less than five answers, we decided not to report the results.

In a second step, we used Split-and-Merge (without the assumption made in Section 5) on the three videos with the following settings: a uniform initial partition with a length of three seconds per block, $\omega_i = \frac{1}{n}$, $\forall i$ (i.e. each feature has the same importance), $\delta = 0.9$ and $\eta = 0.1$. Finally, we reported the partition found under the form of three second intervals centered on the cutting points. The choice to assimilate a cutting point to a three second interval result both from the protocol (people may agree on the same cutting point, but report different time to one or two seconds difference) and from the algorithm (for computational reasons, extracting and processing the screenshots is relatively costly). In the analysis, we consider that two cutting points coincide if there is an intersection between their intervals. In other words, two cutting points are assimilated if they are separated by less than three seconds, which correspond to the selected precision for the algorithm.

6.3 Results

The results are summarized by Figure 2. Let us analyze first the panel answer. For the first video, the cutting points are relatively numerous with 11 in total. A lot of them are mostly in blue indicating a doubt from the panel. Around 1:00, a ten second interval is formed, indicating a variability in the choice of cutting points. This was expected as mentioned previously in the video description. The large block from around 1:00 to 2:25 is also confirmed by the answers. For the second video, and as we expected, there is a neat consensus: every person from the panel answered the same that is to say, putting a cutting point when the transition is visually indicated. There is one exception with a weak transition, in one answer, located on the transition between the black screen at the beginning and the real content of the video. Regarding the third video, the participants seem to agree on whether or not a transition is weak or strong. For the second distinct area in the timeline, around 24s, the union of intervals made of the cutting points has a nine second length. Checking the answer individually confirms that there is no one adding more than one cutting point in this area. The results globally reflect our own personal answers on the cutting points and more important, the panel answers reflect the difficulties we described in the previous section. Globally, the huge amount of weak transitions in the first and last video confirms that the semantic partitioning problem is not well defined as the perception of what exactly is a semantic block varies between persons and videos.

In the first video, only two out of eleven cutting points have been found. There are two pairs of red and blue cutting points separated by only four seconds (in resp. 0:08 and 0.30) and for both, Split-and-Merge identified a cutting point in the middle. We explain this by the fact that the algorithm did not find relevant the blue cutting points (the selected cutting points are closer to the red ones). One good point to notice is the absence of cutting point in the large middle block. In general, the results on this video are disappointing because the video seemed easy to break for humans, and none of the most consensual cutting points (red at 1:10 and 2:28) has been identified. In the second video, the size of the final partition matches the size of the panel optimal partition with seven cutting points. However, except for two points, there are all misplaced: the one around 0:25 is about five second late and the ones 2:20 and 2:40 are three seconds in advance. The two others are clearly misplaced and almost in the worst possible location, i.e. in the middle of an interval (at least not the same). It is quite surprising that the video with the

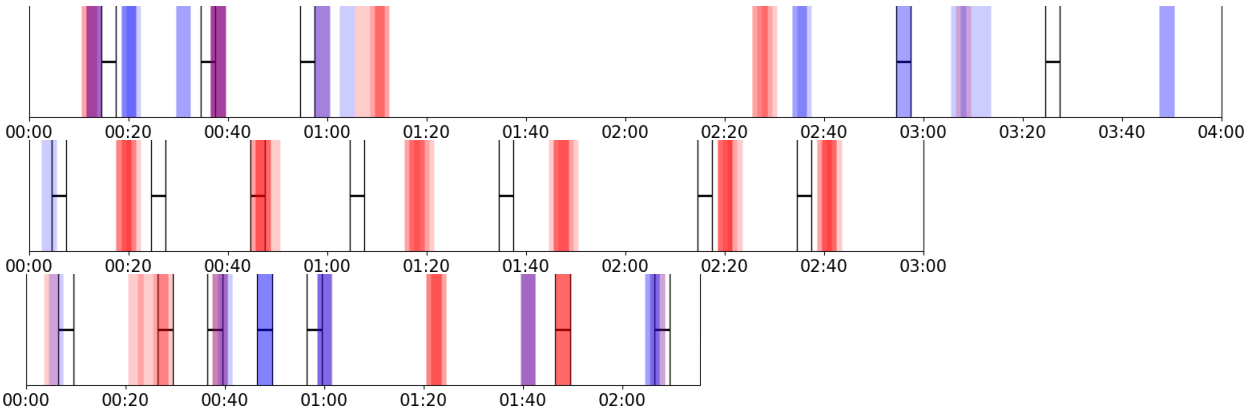

Figure 2: Timelines for the three videos. From the top to the bottom, video 1, 2 and 3. The distribution of *weak* cutting points is indicated in blue, the normal ones in red. A cutting point is materialized by a three second interval and the opacity represents how much an area has been selected as cutting point by the panel.

most clear consensus on the optimal partition does not provide better results. The fact that the speaker appears on the video may be an explanation as between two frames the number of pixels that changed is most likely to be higher without changing the semantics. This would imply that the visual metrics play a more important role than we first expected, at least for videos with this characteristic. Further experiments with different weights ω_i and higher thresholds for the visual metric need to be carried. In the last video, Split-and-Merge successfully identified seven out of nine cutting points among which, two are identified with a zero second difference. However, the most important cutting point (not identified as weak, and being in the five answers with at most one second difference) is not identified. In the large interval around 0:24, Split-and-Merge identified the cutting point at the extreme right side. Surprisingly, Split-and-Merge performed the best on the third video which seems to the authors to be the hardest to naturally break into semantically distinct parts.

6.4 Discussion on the protocol

The small number of answers collected to create the *optimal* partition may be a threat of validity. However, as shown in Figure 2, a consensus seems to emerge, especially for the second video which can mitigate this critic. Another drawback in our protocol is that the results are more qualitative than quantitative: we do not measure any distance from the Split-and-Merge result to the optimal partition. We justify this approach by the fact that such metric may be useful to compare algorithms to each others, but too *ad-hoc* to be interesting by itself. In a preliminary evaluation and taking into account the "fuzziness" of the optimal partition, a qualitative description of the results appeared to be enough to us. A last critic that may appear is the lack of clear definition of what is or should be a cutting point in the instruction for the panel. This has been done on purpose since a more precise indication would probably contain explicit references to some feature streams or metrics on feature streams and indirectly influence the answer.

7 CONCLUSION AND FUTURE WORK

In this paper we presented the Video Semantic Partitioning problem defined as finding an optimal partition w.r.t. an unknown and not so well defined function measuring the *semantic*. We reformulated the problem as the optimization of a convex combination supposedly approximating the unknown function. We justify this construction by the extraction of feature *streams*, each of them holding a part of the semantic contained in the video. To solve the problem, we introduced a heuristic called Split-and-Merge. Its main idea is to cut the video in too many pieces before greedily merging the blocks until the improvement is neglectable. Under a restrictive assumption, we reduced the continuous problem to a (almost) discrete problem and adapted the algorithm consequently. Finally, we presented the results obtained on a few test videos and discuss the protocol.

Despite the variability in the quality of results, the results are encouraging taking into account no hyperparameter tuning was performed. Surprisingly, the algorithm obtains the best results on what we considered to be the hardest video and the worst result on the easiest video.

In future work, we will investigate how to tune the hyperparameters to obtain the best out of the algorithm on a set of videos. We are also interested in comparing the convex combination approach presented in this paper with a multi-objective approach. Also, we need to investigate how good the modified algorithm under the restrictive assumption perform compared to the initial version. Last but not least, we plan to incorporate more advanced feature streamed such as objects obtained by video segmentation.

ACKNOWLEDGMENT

The authors warmly thank IBM BTO and IBM Krakow Software Lab for their financial support.

AUTHOR CONTRIBUTIONS

Marcin Janiszewski pointed out the problem and guided its resolution using IBM Watson Services.
Krzysztof Jamrog implemented the algorithm, *taught* IBM Watson Speech-to-Text to increase the transcript quality.
Alexandre Quemy conceived the algorithm, wrote the article, selected the data, conceived and run the experiments, analyze the data.

REFERENCES

[1] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. 2012. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science* 425, Supplement C (2012), 75 – 103. https://doi.org/10.1016/j.tcs.2011.03.012 Theoretical Foundations of Evolutionary Computation.

[2] Nyein Myint Myint Aung and Su Su Maung. 2013. Semantic Based Text Block Segmentation Using WordNet. *International Journal of Computer and Communication Engineering* 2, 5 (2013), 601.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[4] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. 2005. Evaluating the &Epsi;-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions. *Evol. Comput.* 13, 4 (Dec. 2005), 501–525. https://doi.org/10.1162/106365605774666895

[5] Jisheng Liang, Ihsin T. Phillips, and Robert M. Haralick. 2000. Consistent Partition and Labelling of Text Blocks. *Pattern Anal. Appl.* 3 (2000), 196–208.

[6] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1 (ETMTNLP '02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 63–70. https://doi.org/10.3115/1118108.1118117

[7] Achille Messac, Cyriaque Puemi-Sukam, and Emanuel Melachrinoudis. 2000. Aggregate Objective Functions and Pareto Frontiers: Required Relationships and Practical Implications. *Optimization and Engineering* 1, 2 (01 Jul 2000), 171–188. https://doi.org/10.1023/A:1010035730904

[8] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.