A Lightweight and Efficient Mechanism for Fixing the Synchronization of Misaligned Subtitle Documents

Rodrigo Laiola Guimarães
IBM Research
Rua Tutóia 1157
04007900 São Paulo, Brazil
+55 11 2132 2283
rlaiola@br.ibm.com

Priscilla Avegliano
IBM Research
Rua Tutóia 1157
04007900 São Paulo, Brazil
+55 11 2132 5790
pba@br.ibm.com

Lucas C. Villa Real
IBM Research
Rua Tutóia 1157
04007900 São Paulo, Brazil
+55 11 2132 4548
lucasvr@br.ibm.com

ABSTRACT
Online subtitle databases allow users to easily find subtitle documents in multiple languages for thousands of films and TV series episodes. However, getting the subtitle document that gives satisfactory synchronization on the first attempt is like *hitting the jackpot*. The truth is that this process often involves a lot of trial-and-error because multiple versions of subtitle documents have distinct synchronization references, given that they are targeted at variations of the same audiovisual content. Building on our previous efforts to address this problem, in this paper we formalize and validate a two-phase subtitle synchronization framework. The benefit over current approaches lays in the usage of audio fingerprint annotations generated from the base audio signal as second-level synchronization anchors. This way, we allow the media player to dynamically fix during playback the most common cases of subtitle synchronization misalignment that compromise users' watching experience. Results from our evaluation process indicate that our framework has minimal impact on existing subtitle documents and formats as well as on the playback performance.

CCS Concepts
• Information systems → Multimedia information systems, Speech / audio search • Applied computing → Document management and text processing, Document metadata, Document preparation, Annotation, Format and notation, Multi / mixed media creation.

Keywords
Subtitles; Audio fingerprinting; Synchronization; SRT.

1. INTRODUCTION
Downloading a subtitle document from the Internet and playing it alongside audiovisual content (*e.g.*, a movie or a TV series episode) is not rocket science; but it sure can feel that way sometimes. Considering a user already has the media file on his or her local device and that s/he has identified multiple versions of potential subtitle documents on an online repository, s/he still has to figure out which of such files gives satisfactory synchronization. The problem is that even with the efforts of online communities to review and correct user-contributed subtitle documents as well as media players that try to download suitable subtitle documents automatically, the user may still run endless times into versions that do not sync up perfectly with the base audiovisual content. The underlying problem is that even if the synchronization is off for just a couple of seconds, misaligned subtitle entries will most probably be a constant annoyance.

Take Figure 1 as an example. Here, we illustrate the playback of 2 subtitle documents with the corresponding audiovisual content (track in light blue with dot pattern). Figure 1.a represents the ideal scenario where subtitle entries (in yellow with line pattern) are perfectly synchronized with the base content. On the other hand, in Figure 1.b the timing of all subtitle entries (in orange with line pattern) are shifted ∂t seconds. Note that this latter

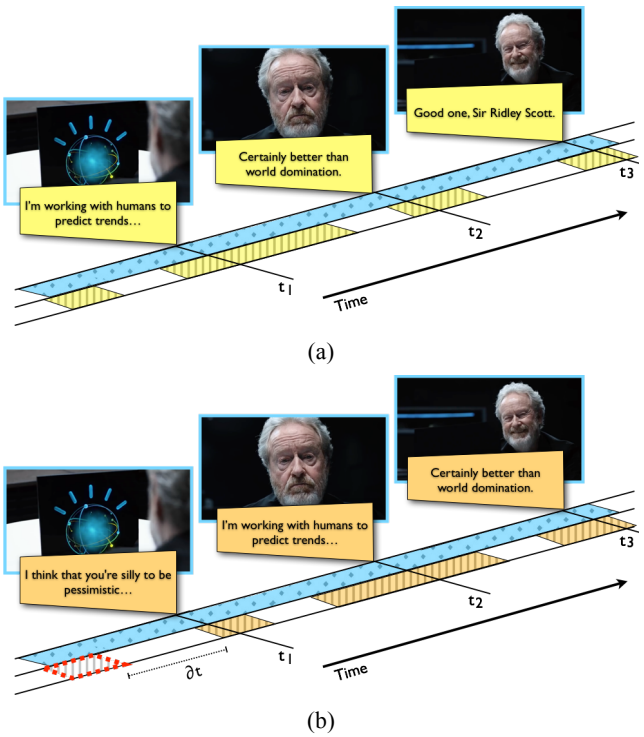

Figure 1. Playback of audiovisual content together with subtitle documents using a local media player: a) subtitle entries with perfect timing and b) shifted ∂t seconds. Screenshots extracted from "Ridley Scott + IBM Watson: A Conversation". Available at https://youtu.be/KDtxQRH8aI4.

DocEng '16, September 12 - 16, 2016, Vienna, Austria.

ACM 978-1-4503-4438-8/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2960811.2960812

subtitle document may have been created using a variation of the original audiovisual content (*e.g.*, a version including advertisements in the beginning). The point is that, although this second document is a potential match, even minor synchronization misalignments might compromise the entire experience. In this context, we consider the scenario in which once a candidate subtitle document has been identified – not necessarily only the one synchronized *to the letter* – the viewer can obtain satisfactory synchronization. Our ultimate goal is to provide the media player the ability to dynamically adjust the presentation of the subtitle document in Figure 1.b, so that the experience from the viewer's perspective looks just like the one in Figure 1.a. By supporting this functionality we expect to minimize the burden on the viewer before the *fun* starts (after all, this is what really matters).

In previous work, we performed a qualitative analysis of several subtitle documents for a popular movie and TV series episode in order to understand the problem domain [20]. This process allowed us to identify some common types of synchronization problems users[1] face when playing audiovisual content together with subtitle documents downloaded from the Internet. To address these issues (*i.e.*, constant and varying temporal offsets) we proposed a two-phase subtitle synchronization mechanism to 1) enrich subtitle documents with audio fingerprint[2] annotations generated from the base audio signal, that later can serve as second-level synchronization anchors for the media player to 2) adjust misaligned subtitle entries during playback.

In this work, we reflect on our previous findings and look at the subtitle misalignment problem from a document engineering perspective. As our first contribution, we formalize a lightweight method that annotates subtitle documents with representative audio fingerprints. We show that the impact of such method to enrich existing subtitle documents and formats is relatively small. As our second contribution, we propose an algorithm that dynamically adjusts the synchronization of misaligned subtitle entries during playback. Experiments with a proof of concept application that realizes the proposed framework indicate that our solution does an efficient use of computational resources.

In particular, the requirements and constraints that motivated our design choices include:

i. *Minimize user effort:* to be practical, the proposed framework must fix the synchronization of misalignment subtitle documents with minimal user input. That seems to make good sense, specially if users spend much more time than necessary in a process that can be automatized;

ii. *Ensure copyright compliance:* the proposed framework should retain the base video integrity, either in terms of editing, removing or adding third-party material to the base audiovisual content, as well as avoid infringing the copyrights in the reuse and reproduction of unauthorized portions of the audio stream;

iii. *Be backward compatible:* a video player that does not implement the proposed framework should process a new version of the subtitle document containing second-level synchronization anchors in the same way as an older version of such document that does not include audio fingerprint annotations. Similarly, new video players should be capable of processing subtitle documents without synchronization anchors;

iv. *Minimize the impact on subtitle documents and formats:* the effect of inserting synchronization anchors based on audio fingerprints should be minimized not only in terms of extending the specification of existing subtitle formats and storage costs, but also in regards with the playback performance; and

v. *Handle different types of subtitle misalignment problems:* the methods to annotate and fix the presentation of subtitle documents enriched with audio fingerprint annotations should be general enough to address the different types of synchronization problems identified in our initial findings.

This paper is organized as follows. In Section 2 we contextualize and motivate our work. Next, in Section 3 we introduce a general framework that addresses the most typical cases of subtitle synchronization misalignment. In brief, the proposed approach consists of 2 steps. Firstly, the enrichment of exiting subtitle documents with representative audio fingerprint annotations generated from the based audio signal; and subsequently, the resynchronization of misaligned subtitle entries during the playback of a variation of the original audiovisual content. Then, Section 4 reports on the design and implementation of a proof of concept that realizes our contributions, whereas Section 5 presents its evaluation process. Finally, Section 6 reviews our contribution in the light of related work and Section 7 is dedicated to concluding remarks and future work.

2. BACKGROUND

The lifecycle of subtitles can be analyzed from different perspectives. In professional post-production, extensive support is typically available for professionals to create and synchronize subtitle entries along with the base audiovisual content. The result from this process can be either *burned-in* in the base content or encapsulated with other data streams (including subtitle documents in other languages) in a container format[3] and distributed on DVDs[4], Blu-Rays, broadcast television, or on-demand video services. Given that all data is wrapped (and distributed) in a single and self-contained unit, synchronization problems[5] like the ones discussed in this paper are not expected during playback.

Non-professionals can also create subtitle documents, although this task can be timing consuming. Sharing the resulting subtitle documents on the Web is particularly common for popular movies and TV series episodes shared online. As these subtitle documents

[1] The terms 'user' and 'viewer' will be used interchangeably in this paper to describe regular people who operate computer software with minimal technical expertise or previous training.

[2] An audio fingerprint is a compact content-based signature that summarizes an audio sample with a predefined length. Especially, it does NOT represent the audio signal at a specific point in time.

[3] Wrapper format whose specification describes how different media types and metadata coexist in a computer file.

[4] Some technologies, if unknown, could easily be identified via an online search; therefore they will not be Web-referenced.

[5] Videos with high resolutions may still not play smoothly on devices with limited computational resources, which may result in a slight synchronization misalignment not only of subtitle entries but also between the audio and video streams.

are often distributed separately from the reference audiovisual content, recipients may face difficulties in finding the version that offers satisfactory synchronization (vide Figure 1). It is important to mention that the existence of multiple versions of subtitle documents may reflect the various editions of the original audiovisual content that may or not include an opening intro, scenes from previous or next episodes, advertisements, and so on. As we will show in the following sections, it is exactly in this scenario that we propose a lightweight and efficient mechanism for fixing automatically the synchronization of misaligned subtitles documents.

2.1 Subtitle Formats on the Web

Bulterman et al. [3] presents an extensive analysis of several subtitle formats, which according to them fall under 2 main categories: embedded and external text formats. Embedded text formats are tightly integrated with the host language. One example of said format is SmilText, which contains intra-block formatting and timing control, with the layout and general rendering control defined in the Synchronized Multimedia Integration Language (SMIL) [4].

External subtitle formats, on the other hand, encapsulate information on synchronization and text styling in an external file or document. Once in the possession of such document, the media player parses its contents and renders the text entries on the screen according to the timestamps and durations specified for each entry. Below, we present some external subtitle formats that are prominent on the Web.

WebVTT[6] (*Web Video Text Tracks Format*) is a technology introduced in HTML5 that can be used to display timed text tracks with the HTML5 `<track>` element alongside `<video>` elements. The file is text-based and contains essentially (i) a header, (ii) a sequence of subtitle entries, and (iii) empty lines. Subtitle entries have a numerical identification, starting and ending times, and a textual payload. This format also admits text-formatting settings such as text direction (*e.g.*, left to right or the other way round), rendering position, text size and alignment, and bold/italic/underlined tags, to name a few. WebVTT also supports comments by starting a line with the string "NOTE". Said lines are not rendered on the screen.

```
01. ...
02. 19
03. 00:01:22,782 --> 00:01:27,221
04. RIDLEY SCOTT: I think that you're silly to be
05. pessimistic...
06.
07. 20
08. 00:01:29,467 --> 00:01:34,172
09. WATSON: I'm working with humans to predict trends...
10.
11. 21
12. 00:01:35,524 --> 00:01:37,248
13. RIDLEY SCOTT: Certainly better than world
14. domination.
15.
16. 22
17. 00:01:37,688 --> 00:01:39,053
18. WATSON: Good one, Sir Ridley Scott.
19. ...
```

Figure 2. Typical SRT document shared online. Each subtitle entry includes a sequence number (in black), the time the text should appear and disappear (in red, left and right side of the '-->' token, respectively), and the textual content (in blue).

[6] https://www.w3.org/TR/webvtt1/

SubStation Alpha[7] (SSA) is a popular file format used in conjunction with the Matroska MKV container to store subtitle data along with video streams. The structure of an SSA document is similar to an INI file: sections are declared with `[brackets]`, lines starting with a semicolon (`;`) are treated as comments, and pairs of `key:values` are used to define subtitle metadata and attributes. Attributes include formatting and styling, scaling, rotation, and font names, among others. Actual subtitle entries are described in an `[Events]` section that includes not only the textual payload but also the position of the text, timestamps, and effects. The format is rich in the sense that it enables the creation of complex presentations, although the resulting subtitle document file may not be adequate to be manually edited in a text editor due to its size and relative complexity.

The most popular of the external text-based file formats in use today is, by far, the SRT (*SubRip Text*). SRT is widely supported by both a variety of players and subtitle creation programs. Essentially, a SRT document contains the textual entries to be displayed and the moment of that presentation. There is no support for comments. Some media players recognize text-formatting (bold, italic, underline, and font color) commands entered with HTML tags. As illustrated in Figure 2, SRT is comprised of three main elements: a sequence number (lines 02, 07, 11 and 16), the time in which the subtitle must appear and disappear on the screen (lines 03, 08, 12 and 17), and the subtitle text itself across one or more lines (lines 04-05, 09, 13-14 and 18).

2.2 Preliminary Findings

To identify common synchronization misalignment problems, in previous work [20] we analyzed multiple versions of SRT documents related to a highly rated movie and TV series episode – according to the Internet Movie Database (IMDb). All documents were obtained programmatically using the open API (*Application Program Interface*) available on OpenSubTitles.org. The rationale to analyze subtitle documents related to 2 different productions was that in essence movies have a different structure when compared to TV series. For instance, a TV series episode may start with an opening intro or scenes from the previous episode, and finish with a preview of the next episode.

For the referred movie, we obtained 193 SRT documents distributed across 33 different languages (English, Brazilian Portuguese, and Spanish were among the most frequents). In general, these subtitle documents included advertisements and credits information as actual subtitle entries. Interestingly, we noticed that in several opportunities the very same subtitle document had different file names and different creators listed in the credits. This suggests that ownership infringement is a recurrent problem in subtitle sharing communities on the Internet.

In our analysis for the episode of a popular TV series, we collected 170 SRT documents in 38 different languages. In contrast to the ones for the movie, the presence and absence of prologues and epilogues was often observed and caused a relatively large standard deviation at the presentation time of the first subtitle entry (~45 seconds for an average of 88 seconds in the case of the subtitle documents for Brazilian Portuguese).

In our analysis, we also found that some SRT documents are extracted directly from DVDs and Blu-Rays, whereas others are created using audiovisual content recorded from broadcast TV. More importantly, we noticed that the main synchronization

[7] https://www.matroska.org/technical/specs/subtitles/ssa.html

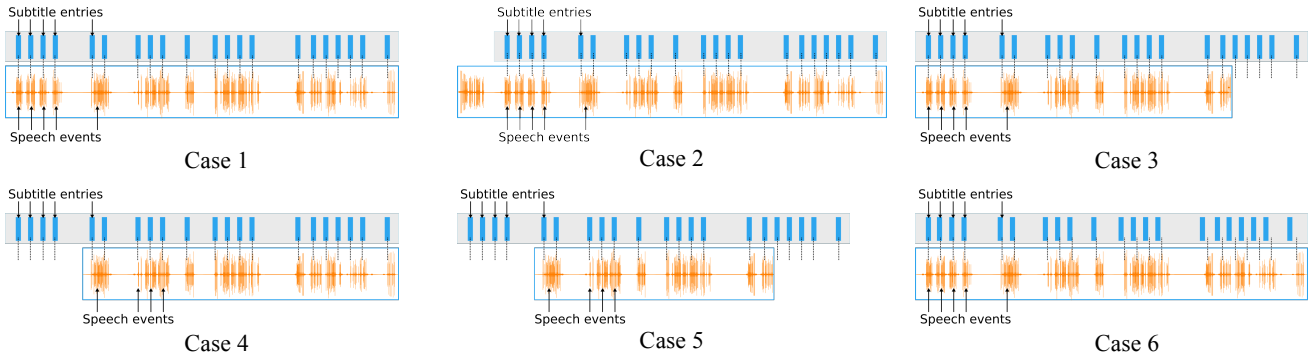

Figure 3. Alignment scenarios when playing audiovisual content together with subtitle documents downloaded from the Internet.

problem between different subtitle documents is a *constant temporal offset* that shifts equally the presentation of equivalent subtitle entries (as illustrated in Figure 1). This is the case, for instance, if one SRT document is generated using an audiovisual content that has a prologue; whereas another document is created using a version of the same audiovisual content that does not include such part.

To a lesser extent, we also observed that some subtitle documents have a *varying temporal offset*. In this case, even when different documents have the first subtitle entry temporally aligned, the next entries increasingly get out of sync with each other. In other words, it is like if the ∂t in Figure 1.b started equals zero but increased over time. The causes of this problem suggest that varying temporal offsets are related to different encoding offsets (*e.g.*, frame rate) used in the corresponding base contents.

Analyzing the collected data, we grouped the synchronization misalignment problems between audiovisual content and subtitle documents in 6 canonical cases[8], as illustrated in Figure 3:

- *Case 1 - Perfectly synchronized:* the audiovisual content and the subtitle document are in sync, or in other words, the offset ∂t is equal to zero;
- *Case 2 - Audiovisual content includes an initial part:* such part did not exist in the version used to generate the subtitle document. In this case, during playback all subtitle entries should be shifted an offset ∂t greater than zero;
- *Case 3 - Audiovisual content trimmed in the end:* the offset ∂t is equal to zero, but the subtitle document has entries that do not have counterparts on the audiovisual content (these are never exhibited during playback by the way);
- *Case 4 - Audiovisual content trimmed in the beginning:* the initial subtitle entries in the document are not presented; hence, the value of the offset ∂t is negative;
- *Case 5 - Audiovisual content trimmed on both ends:* only a subset of the entries in the subtitle document should be presented. As the initial entries would be discarded, the offset ∂t should have a negative value;
- *Case 6 - Audiovisual content with different pace:* this is a peculiar situation, possibly generated when the media file was encoded in another format. The subtitle entries start with a given ∂t, but as time passes by, this value increases/decreases.

[8] Naturally, a combination of these canonical cases may apply in some scenarios.

3. GENERAL FRAMEWORK

The study presented in the previous section reinforces the argument that finding a subtitle document that offers users satisfactory synchronization is a very challenging task. The several synchronization misalignment variations identified also support our premise that a mechanism to automate the synchronization of subtitle documents during playback is indeed relevant and necessary. Therefore, to address this research problem we propose a method that (1) enriches subtitle documents with representative audio fingerprint annotations extracted from the base audio signal during the authoring process and (2) adjusts misaligned subtitle entries based on the comparison of audio fingerprints during playback.

3.1 Enriching Subtitle Documents

The first step of our framework consists in automatically annotating subtitle documents with representative audio fingerprints, as illustrated in Figure 4.a. To do that, the authoring software first extracts the audio signal from the base audiovisual content. Then, it processes the extracted audio signal and generates a number of audio fingerprints that are later encapsulated, preferably as metadata, in the subtitle document. As we will see ahead, such audio fingerprint annotations also include the corresponding offset within the base audiovisual content, so this information can be used during playback as second-level synchronization anchors to fix the presentation of misaligned subtitle entries.

To encompass all the 6 cases listed in Section 2.2, we make use of 3 synchronization anchors. The rationale behind this design choice is the following. In fact, 2 synchronization anchors would be enough to handle all the problems. However, as the audiovisual content may have been trimmed in the beginning or in the end, we propose the insertion of a fallback anchor. As a guideline, the authoring software should extract three audio fingerprints preferably near the beginning, near the middle, and close to the end of the audio signal and then insert such fingerprints with the corresponding offsets as annotations in the subtitle document. It is worth mentioning that audio fingerprint annotations do not necessarily need to be associated with an actual speech event.

3.2 Adjusting Misaligned Subtitle Entries

The second step of our framework takes place during the playback of an enriched subtitle document along with an audiovisual

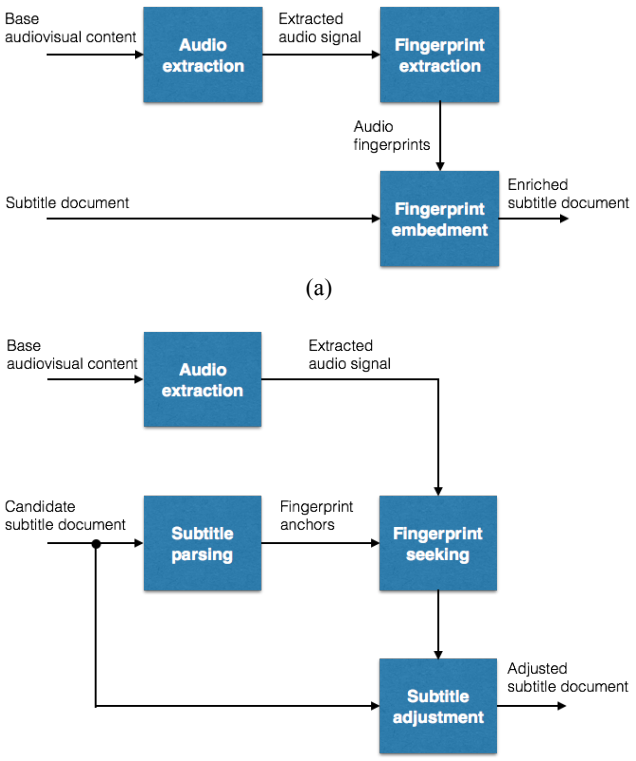

Figure 4. General framework in two steps: (a) enrichment of subtitle document with representative audio fingerprint annotations and (b) dynamic adjustment of subtitle entries based on fingerprints comparison.

content (see Figure 4.b). The process begins with the media player opening the base audiovisual content and extracting the audio signal. Complementary, the media player parses the subtitle document, from where subtitle entries and audio fingerprint annotations are read. Then, our algorithm, as illustrated in the simplified pseudocode procedure of Algorithm 1, seeks the synchronization anchors (denoted as the *fingerprints* array of length 3 and its associated *timestamps* array indicating where the fingerprints start) in the extracted audio signal (denoted as the *audio* array with the decoded signal). This algorithm returns an array containing three possible values for each fingerprint comparison: `NO_MATCH`, `MATCH`, or `LOCAL_MATCH`.

Algorithm 1 Fingerprint seeking algorithm

```
01. function seek(fingerprints, timestamps, audio):
02.   match = [NO_MATCH, NO_MATCH, NO_MATCH]
03.   for i = 1 to 3:
04.     nsecs = DEFAULT_FP_DURATION /* in secs */
05.     seek_offset = timestamps[i]
06.     for attempt = 1 to 2:
07.       test = captureFP(audio, seek_offset, nsecs)
08.       if test.matches(fingerprints[i]):
09.         delta = test.matchOffset()
10.         if delta != 0: /* propagate delta */
11.           for j=i+1 to 3: timestamps[j] += delta
12.         match[i] = attempt==1? MATCH : LOCAL_MATCH
13.         break
14.       else:
15.         nsecs = LOCAL_SEARCH_INTERVAL /* in secs */
16.         seek_offset = MAX(seek_offset - nsecs/2, 0)
17.   return match
```

Algorithm 2 Subtitle adjustment algorithm

```
01. /* actions variable corresponds to Table 1 */
02. function adjust(matches, actions):
03.   for i=1 to rows(actions):
04.     if matches[1] == actions[i].FirstAnchor and
05.        matches[2] == actions[i].MiddleAnchor and
06.        matches[3] == actions[i].LastAnchor:
07.       Process action at actions[i].Action
08.       return
```

First, our algorithm seeks to the exact offset described by each synchronization anchor and generate an audio fingerprint from the audio signal (line 07); if there is a match in the first attempt (line 08), the given synchronization anchor is indeed aligned (`delta` variable is set to zero in line 09). Algorithm 1 then updates `match[i]` with `MATCH` (line 12) and proceeds to the next anchor. In case the audio fingerprint annotation does not match the calculated fingerprint at the expected offset, our method increases the size of the search window (lines 15-16) and performs a new search (`attempt` variable equals 2). This new search consists in generating all audio fingerprints in a predetermined timeframe around the given offset and comparing each one of them with the referred audio fingerprint (local search). If the algorithm successfully finds a match, it updates `match[i]` with `LOCAL_MATCH` (line 12) and proceeds to the next anchor. Otherwise, `match[i]` keeps the initial value (`NO_MATCH`).

The candidate subtitle document is perfectly synchronized with the base audiovisual content only when the media player directly finds matches for all synchronization anchors in the first attempt. In case the media player performs a local search to find a match (`attempt` variable equals 2), the difference between the offset specified in the synchronization anchor and the one calculated from the audio signal is taken into account in the verification of the coming anchors (lines 09-11 in Algorithm 1). Therefore, even if the first synchronization anchor is found through a local search, the next anchors can still match perfectly if such difference of offsets is considered in the calculations ahead.

In most cases, the adjustment action consists in propagating a constant temporal offset (or `delta`) to all the subtitle entries specified in the subtitle document. However, when the media player finds two synchronization anchors via local searches, it is the case in which the subtitle document and the base audiovisual content have different presentation paces (varying temporal offset). In this scenario, the player could interpolate the difference between the offsets specified and calculated so that such differences are distributed accordingly to all the subtitle entries before, in between and after the synchronization anchors. Table 1 summarizes a naïve algorithmic procedure (Algorithm 2) and adjustment actions for all the 6 cases presented in Section 2.2.

4. IMPLEMENTATION

We prototyped a proof of concept to investigate both the impact of inserting audio fingerprint annotations into preexisting subtitle documents and the performance implications associated with dynamic adjustment of misaligned subtitle entries during playback. For that we used the well-known VLC media player and Chromaprint[9], a client-side open source library written in C implementing a custom algorithm for extracting audio fingerprints from raw uncompressed audio data sources.

[9] http://acoustid.org/chromaprint

Table 1. Adjustment actions in different scenarios.

First anchor	Middle anchor	Last anchor	Case	Action
Match	-	Match	Case 1	None
Match	-	Local match	Case 6	Interpolation
Match	Match	-	Case 3	None
Match	Local match	-	Case 6	Interpolation
Local match	-	Match	Case 2	Offset propagation
Local match	-	Local match	Case 6	Interpolation
Local match	Match	-	Case 5	Offset propagation
Local match	Local match	-	Case 6	Interpolation
-	Local match	Match	Case 4	Offset propagation
-	Local match	Local match	Case 6	Interpolation
-	Local match	-	Case 5	Offset propagation
-	-	Local match	Case 5	Offset propagation
-	-	-	-	None

Note: the symbol '-' applies either when a synchronization anchor is not found or when there is no need to use it.

The Chromaprint library works with *spectrograms*, which are visual representations of the spectrum of frequencies in a sound as these vary with time. Spectrograms can be calculated from splitting the original audio into many overlapping frames and then applying a Fast-Fourier transform (FFT) on them. In particular, Chromaprint converts the input audio to the sampling rate of 11025Hz and using a FFT window size of 4096 (0.371s) with 2/3 overlap. It further processes the information by using a Short-time Fourier transform and by converting frequencies into musical notes. The result, which has 12 bins (one for each semitone of a chromatic scale), is known as *chroma features* [1]. It is worth mentioning that this representation of the audio is not radically affected by differences between codecs, and more importantly, it can be used to measure the similarity with other representations (*e.g.*, by calculating bit error rates). By moving a prefixed sliding window over the spectrogram representation of the audio from the left to the right, one pixel at a time, we can generate several sub-images of that spectrogram. On each of the sub-images, Chromaprint applies a pre-defined set of 16 filters that capture intensity differences across musical notes and time (encoded into 2 bits for each filter using the Gray code). Following the same process for each and every single sub-image, Chromaprint obtains the full audio fingerprint.

Chromaprint needs about 3 seconds of audio samples to fill the library's internal buffers; consequently, a larger number of samples are needed to generate enough sub-images. In our prototype, we chose to capture 30 seconds of audio (herein defined as *N*), which is sufficient to represent unique sequences of audio events in a movie or TV series episode. To compute the correlation between the captured and the reference fingerprints, we check how many bit differences there are between the two. A moving window accounts for temporal misalignment. As output, we have the fingerprint offset from where the two fingerprints match the best and how similar they are (*i.e.*, a correlation "score"). If the score obtained is greater than a given threshold, we assume that the compared fingerprints match.

To deal with situations in which things like prologues and epilogues are included or removed, we capture audio fingerprints using a different time length every time the player does not find a direct match, and therefore a broader local search is needed. For local searches, we pick the standard deviation *D* presented in Section 2.2 and make our sample size *D*+*N*+*D*.

In our prototype we chose to support the popular SRT subtitle format, as shown in Figure 5. In order to keep backwards compatibility with media players that do not include support for audio fingerprint annotations, we decided to encapsulate synchronization anchors as subtitle entries with zero duration (note the starting and ending times in line 8). The keyword `@fingerprint@` (line 09) indicates the presence of the audio signature introduced by our annotation technique. Note that such information is stored in the same area that SRT reserves for the text that must be exhibited on the screen.

5. EVALUATION

We conducted two studies to validate the core contribution of our work. In the first, we analyzed the impact of inserting audio fingerprint annotations into subtitle documents; whereas in the second, we estimated the computational resources demanded to generate audio fingerprints and to run the correlation routines that detect matches between the reference and captured audio samples.

The impact on the document size is measured by the amount of text introduced with the insertion of the audio fingerprint annotation in the SRT file. An audio fingerprint is nothing more than an array of integers – and that is the representation we use in our software when we compute the correlation between two audio signatures. When it comes to storing that information on the subtitle document, however, that representation is not the most adequate; it is both meaningless to users who inspect the subtitle document as well as potentially long in number of characters (spanning several lines in the document). For such reasons, we encode the audio fingerprint as a Base64 string in the document.

```
01. ...
02. 19
03. 00:01:22,782 --> 00:01:27,221
04. RIDLEY SCOTT: I think that you're silly to be
05. pessimistic...
06.
07. 20
08. 00:01:28,000 --> 00:01:28,000
09. @fingerprint@ AQAAjFEiSYmSJJGkAOKP7_hx4...
10.
11. 21
12. 00:01:29,467 --> 00:01:34,172
13. WATSON: I'm working with humans to predict trends...
14.
15. 22
16. 00:01:35,524 --> 00:01:37,248
17. RIDLEY SCOTT: Certainly better than world
18. domination.
19.
20. 23
21. 00:01:37,688 --> 00:01:39,053
22. WATSON: Good one, Sir Ridley Scott.
23. ...
```

Figure 5. SRT document including an audio fingerprint annotation, where keyword `@fingerprint@` is followed by a Base64 string representation of such audio signal's signature.

The length of the Base64 string varies according to the value of the elements of the original array. As noticed in the process of annotating 10 movies, the average size of one encoded audio fingerprint was 247±4 ASCII characters. Therefore, when we consider all 3 fingerprints captured along with the text that describes their zero-duration timestamps and sequence numbers, the average overhead added to each document is around 890 characters (or bytes). To put in perspective, in a regular movie with about 1000 subtitle entries, these annotations would represent just about 1% of the file size.

Next, we analyzed the amount of computational resources demanded by our method. We chose a high-definition movie featuring a H.264 video stream with 1920x800 resolution and an AAC (LC) audio stream, and then observed CPU and memory usage during a timespan of 120 seconds through a series of 10 runs. On the hardware side, the tests were performed on a dual-core Intel i7-3520M running at 2.9GHz and with 6GB of main memory. The software stack included our prototype (linked to Chromaprint version 1.3.1 and to VLC media player library version 2.2.1) and an operating system based on Linux 4.5. In order to optimize cache performance, CPU affinity was configured so that our prototype was bounded to the same processing unit during its execution.

The percentage of CPU demanded to decode the video stream (plotted with red filled squares in Figure 6) varies between 15% and 48% for the short segment of the movie selected. This variation reflects the complexity of the scenes; the smaller the differences between a frame X and a frame X+1 the lesser computing resources are used. Decoding the audio stream is a much cheaper task, as the line with hollow circles shows (in blue). In the moments filled with rich sounds, CPU usage goes up to 6%. In a second moment, when music becomes less complex, audio decoding demands drops to about 3% of processing power. Standard deviation was not statistically significant, and therefore it is not plotted in the figure.

In the lower part of Figure 6, the line with filled circles (in purple) shows the percentage of CPU required to process the audio fingerprints. The fingerprinting process begins with the aggregation of audio packets into a buffer. The cost of that task is very low and remains at 1-2% at all times. Once enough data has been aggregated, Chromaprint computes the fingerprint of that buffer and the result is compared with the reference fingerprint (stored in the annotated SRT document). This process produces the peaks at 18 seconds (with 18% of CPU consumption) and at 108 seconds (with 35% of CPU consumption) and lasts no more than 125 milliseconds in our setup.

The difference between the two purple peaks is that in the first we look for a perfect match between the reference and the computed fingerprints. In other words, both audio fingerprints have the same number of elements and represent the same amount of time. The second peak shows the CPU demanded in a local search. In this case, the aggregated buffer includes 45 extra seconds of sampling before and after the reference timestamp of the fingerprint. Because of this larger buffer size, the fingerprint comparison demands the use of a sliding window and, as a consequence, the processing power needed exceeds that of the former case.

Nevertheless, we note that this is a one-time task that can be performed before the media file starts to be reproduced to the user. Also, we note that media parsing for fingerprinting computation purposes does not need to follow the presentation timestamps (PTS). Consequently, users need to wait for no more than a few hundred milliseconds before the subtitle document is processed and adjusted (if it really needs be).

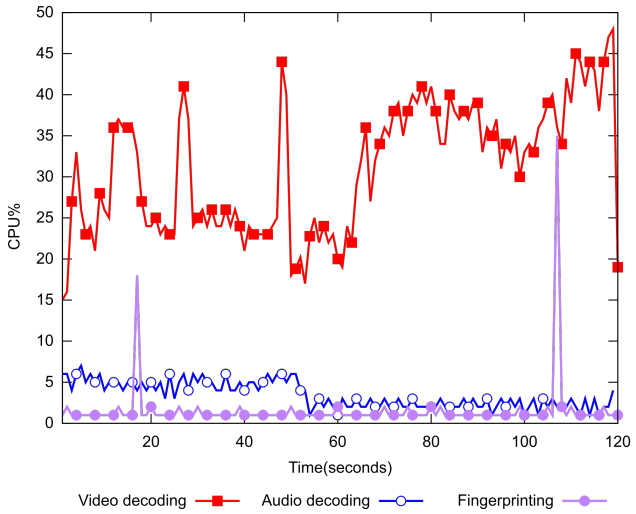

Figure 6. Processing power demands to process audio fingerprints and to decode video and audio streams.

The impact on memory consumption (not shown in the figure) is also relatively small, as all it takes is a few seconds of audio samples in memory to buffer and capture the audio fingerprint. When computing a local search (which demands more memory than a perfect match), the amount of memory used by our software is of no more than 4MB. In comparison, the audio software decoding process has a baseline of 43MB. At its peak, memory overhead relative to the baseline stayed at 8,5%. Such a small footprint enables the use of our technique even in embedded devices with limited amount of memory. In such a scenario, we note that memory requirements can be further reduced by tuning the local search window to smaller values.

6. RELATED WORK

Subtitles play an important role in making audiovisual content accessible to everyone. For instance, it is often the case in which subtitles are necessary to watch a movie or TV show in a noisy environment (*e.g.*, in an airplane) or when one is not familiar with the language or accent in the audio streams. The truth is that the benefits of subtitling go beyond speech information and description of representative events in the audiovisual content. In the literature, extant research has investigated the impact of subtitles in terms of accessibility [11], cognitive load [14][15], comprehension of foreign languages [17][18] and vocabulary learning [13], to name a few. In the remaining of this section, we review some representative efforts in the context of our work.

Hong et al. [11] propose a dynamic captioning[10] approach, which explores a set of technologies including face detection and recognition, visual saliency analysis and text-speech alignment. They investigate whether subtitles placed at suitable positions help hearing-impaired people recognize speaking characters and perceive the moods that are conveyed by the variation of volume. Complementary, Wang et al. [21] propose a method to enrich the visualization of videos with visual representation of non-verbal

[10] Although the terms 'captions' and 'subtitles' do have different meanings in some countries, such distinction is not relevant in the context of this paper.

sounds. Their approach automatically transforms non-verbal video sounds into animated words, and positions these near the sound source objects in the video. The dynamics of the animation is based on the intensification and attenuation of the sound volume, whereas the animation positioning is computed using a 3D video cost field of the input video depending on the position of the sound source object. Yet in another effort, Brown et al. [2] use eye-tracking data to investigate the effect of dynamic subtitles in the viewing experience of subjects with hearing loss. Finally, Hughes et al. [12] propose the use of responsive Web design practices to rendering subtitles alongside video content. The proposed approach interpolates individual word timings based on each word's position in the subtitle entry and the start and end time of the entry. The authors also consider the use of phonetic models and semantic markup to dynamically re-block subtitles as a response to user interaction. Responsive subtitles are then formatted and displayed appropriately for different devices while respecting the requirements and preferences of the viewer. Our work is related to these efforts, but instead we focus on a different core technology and application domain.

Liu and Wang [16] propose a stroke-like edge detection method based on contours to extract captions that are hard-coded in videos. Instead of regarding each video frame as an independent image, the authors demonstrate that the use of inter-frame information can improve the accuracy of caption localization and segmentation. Our work differs not only in the use of technology, but also in the underlying research problem, that in our case is automatically fixing the synchronization of subtitles entries encoded in an independent subtitle document; so that viewers can still obtain a satisfactory watching experience.

Fererico and Furini [8] propose a caption alignment mechanism that exploits common off-the-shelf automatic speech recognition (ASR) applications to produce time-coded transcripts. Their approach does not require human transcriptions or special dedicated software. They introduce a unique audio markup into the audio stream before passing it to an ASR application. By knowing the temporal locations of the inserted audio markups, the mechanism can automatically transform the plain transcript produced by the ASR application into a time-coded transcript. Similarly, the video-sharing service YouTube™ uses ASR technology to automatically generate, synchronize and translate captions for videos users upload. In a typical (automated) subtitling process, the original speech is first translated fully into the target language and then the target translation is compressed to optimize the length requirements. One of the techniques employed in the text compression phase is to replace a target language word in the original translation with a shorter synonym of it, thus reducing the character length of the subtitle [9]. Although our work shares the same goal for having subtitles aligned with audiovisual content, we address a different research challenge that is adjusting the synchronization of potential subtitle documents with displaced timestamps.

Tiedemann [19] addresses the particular problem of synchronizing movie subtitles to improve alignment quality when building a parallel corpus out of translated subtitles. In the proposed approach, anchor points are identified based on cognate filters, which use string similarity measures and some heuristics for selecting synchronization. Complementary, the author proposes a dictionary-based approach using automatic word alignment and shows an improvement in alignment quality even for related languages compared to the cognate-based approach. In the context of our work, one could perhaps use a similar approach to adjust the temporal offset of misaligned subtitles. But for that, it would be necessary to know the reference subtitle document that perfectly synchronizes with the base audiovisual content at playback time; what takes us back to *ground zero*.

From a more document engineering perspective, Concolato et al. [6] discuss the synchronized playback of live video and subtitle content using HTTP (*Hypertext Transfer Protocol*) streaming technologies such as MPEG Dynamic Adaptive Streaming over HTTP (DASH). Furthermore, Guimarães et al. [10] describe a set of temporal transformations for multimedia documents that allow users to create and share personalized timed-text comments on third party videos; whereas Fagá Jr. et al. [7] present a vocabulary proposal for third-party applications that allow users to add more generic multimedia annotations to user-generated video content. Most closely related to our work, Bulterman et al. [3] survey many open and proprietary formats for encoding subtitles. Based on a careful analysis, the authors describe a timed-text format that balances the need for style formatting with the requirement for more structured representation that can be easily parsed and scheduled at runtime. Our work builds on these previous findings; however, we go a step further by proposing a framework capable of adjusting the synchronization of tens (or even hundreds) of subtitle documents with displaced timestamps.

Regarding applications, some popular media players offer users a mean to download SRT documents automatically once the base audiovisual content is loaded. For instance, VLC can use an extension called VLSub[11] to search online for the corresponding subtitle document using two different approaches. In the first, the extension uses the media file name to query a remote subtitle database. If there is a match, the corresponding subtitle document is automatically downloaded and presented alongside the audiovisual content. In the second approach, VLSub computes a hash (checksum) of the media file and uses the resulting hash to query the remote subtitle database. Again, if there is a match, the player automatically downloads and displays the corresponding subtitle document. In both cases, the synchronization of subtitle entries might still appear misplaced during playback because: (1) multiple versions of subtitle documents may have the same file name; (2) the based audiovisual content no longer has its original file name due to name mangling or to renaming; or (3) changes to the original encoding settings, such as exporting an original file in H.264 format to a QuickTime .MOV format, will alter the hash of the base media file.

In addition, some media players also allow users to delay or speed up the presentation of subtitle entries during playback, so that users can try to fix synchronization misalignment interactively. Unfortunately, this workaround might require a lot of iterations and yet does not solve the problem once and for all. In this work, we envision a less intrusive mechanism for fixing subtitle synchronization problems without the viewer being even aware that such issues exist. As we demonstrated in the previous sections, this becomes possible through the extraction of audio fingerprints, which is not likely to change drastically when the original audio signal is re-encoded with different settings. Thus, we allow compliant media players to automatically fix the subtitle misalignment problem by considering the audio fingerprint annotations as second-level synchronization anchors. Once such audio fingerprints are identified in a subtitle document, the player

[11] https://github.com/exebetche/vlsub

can then calculate and adjust the temporal offset of all the subtitles entries accordingly.

7. DISCUSSION AND FINAL REMARKS

In this paper we formalized and evaluated a two-phase subtitle synchronization framework that uses audio fingerprint annotations extracted from the base audio signal as second-level synchronization anchors. This approach allows compliant media players to automatically fix (requirement i) common cases of subtitle synchronization misalignment (requirement v) that compromise users' watching experience during playback. Our experiments also show that the overhead introduced by our framework is minimum in terms of document length (*i.e.*, a few extra bytes for the audio fingerprint annotations) and of CPU time/cycles required to process fingerprints (requirement iv).

Because the duration of audio fingerprint annotations is equal to zero, media players that do not implement the proposed alignment mechanism simply do not show those subtitle entries on the screen during playback. This way, we assure that the enriched subtitle documents remains backward compliant and renders the same on the screen as non-annotated subtitle documents (requirement iii).

As we have discussed in the introduction of this paper, we took special care to ensure that the fingerprint representation stored in the subtitle document is non-invertible to the original waveform as a way to protect against the creation of derivative work – in this case, of the original audio signal (requirement ii). We note, however, that the definition of what can be considered an extension of the original work may not even be a concern depending on the license under which the original work is published. That is often the case with the Creative Commons license, popular among artists and independent producers. On the other hand, an in-depth discussion on the legal issues involved in the process of enriching subtitles of commercial movies or of movies that do not feature support for certain languages deserves a separate paper on its own.

One can still argue that while a few years ago the automatic production of a video transcript was very hard to achieve, currently, thanks to the advances in speech technologies, generic off-the-shelf automatic speech recognition (ASR) applications produce reasonable textual versions starting from audio streams. Moreover, this process could perhaps be further improved by considering cognitive computing technology (like IBM Watson in Figure 1) and historic data from featured characters. Not to mention that automatic generation of subtitle entries by services like YouTube and by the reach of Netflix and Amazon Prime, may lead to questions about the relevancy of a technique to synchronize offline media. However, estimates are that in the next 5 years video traffic, *including* peer-to-peer, will be responsible for 80% of all consumer Internet traffic [5]. And combined with the facts that online databases of subtitle documents on the Internet keep growing on a daily basis and that not all movies ever produced are in the catalogs of the aforementioned services, our work is yet relevant and timely.

As future work, we intend to deepen the investigation on the causes of the discrepancies between subtitle documents and on streaming delivery support, which would likely preclude the need for downloading the entire video. These aspects will surely require new enhancements and extensions to our current framework. Moreover, we would like to conduct an experiment to measure the sensitivity of users to the lack of subtitles synchronization and another to assess our method on platforms with limited resources such as small set-tops, tablets etc. This information could be useful for the calculation of the optimized position of anchor points in subtitle documents. We also understand that subtitle formats other than the SRT could also benefit from our work. Thus, a natural sequence to this study would be to conduct a more extensive analysis on a larger corpus of movies and of file formats.

8. REFERENCES

[1] Bartsch, M. A. and Wakefield, G. H. 2005. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*. 7, 1 (Feb. 2005), 96-104. DOI=http://dx.doi.org/10.1109/TMM.2004.840597

[2] Brown, A., Jones, R., Crabb, M., Sandford, J., Brooks, M., Armstrong, M., and Jay, C. 2015. Dynamic Subtitles: The User Experience. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video* (TVX '15). ACM, New York, NY, USA, 103-112. DOI=http://dx.doi.org/10.1145/2745197.2745204

[3] Bulterman, D. C. A., Jansen, J, Cesar, P., and Cruz-Lara, S. 2007. An efficient, streamable text format for multimedia captions and subtitles. In *Proceedings of the 2007 ACM symposium on Document engineering* (DocEng '07). ACM, New York, NY, USA, 101-110. DOI=http://dx.doi.org/10.1145/1284420.1284451

[4] Bulterman, D. C. A. and Rutledge, L. W. 2008. *SMIL3.0 – Flexible Multimedia for Web, Mobile Devices and DAISY Talking Books*. (2nd ed.). Springer Publishing Company, Incorporated. ISBN: 978-3-540-78546-0

[5] Cisco. 2015. Cisco Visual Networking Index: Forecast and Methodology, 2014-2019. *White paper*. Available at http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[6] Concolato, C. and Le Feuvre, J. 2013. Live HTTP streaming of video and subtitles within a browser. In *Proceedings of the 4th ACM Multimedia Systems Conference* (MMSys '13). ACM, New York, NY, USA, 146-150. DOI=http://dx.doi.org/10.1145/2483977.2483997

[7] Fagá Jr., R., Motti, V. G., Cattelan, R. G., Teixeira, C. A. C., and Pimentel, M. G. C. 2010. A social approach to authoring media annotations. In *Proceedings of the 10th ACM symposium on Document engineering* (DocEng '10). ACM, New York, NY, USA, 17-26. DOI=http://dx.doi.org/10.1145/1860559.1860566

[8] Federico, M. and Furini, M. 2014. An automatic caption alignment mechanism for off-the-shelf speech recognition technologies. *Multimedia Tools Appl.* 72, 1 (Sep. 2014), 21-40. DOI=http://dx.doi.org/10.1007/s11042-012-1318-3

[9] Glickman, O., Dagan, I., Keller, M., Bengio, S., and Daelemans, W. 2006. Investigating lexical substitution scoring for subtitle generation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning* (CoNLL-X '06). Association for Computational Linguistics, Stroudsburg, PA, USA, 45-52.

[10] Guimarães, R. L., Cesar, P., and Bulterman, D. C. A. 2010. Creating and sharing personalized time-based annotations of videos on the web. In *Proceedings of the 10th ACM symposium on Document engineering* (DocEng '10). ACM,

New York, NY, USA, 27-36. DOI=http://dx.doi.org/10.1145/1860559.1860567

[11] Hong, R., Wang, M. Yuan, X., Xu, M. Jiang, J., Yan, S., and Chua, T. 2011. Video accessibility enhancement for hearing-impaired users. *ACM Trans. Multimedia Comput. Commun. Appl.* 7S, 1, Article 24 (Nov. 2011), 19 pages. DOI=http://dx.doi.org/10.1145/2037676.2037681

[12] Hughes, C. J., Armstrong, M., Jones, R., and Crabb, M. 2015. Responsive design for personalised subtitles. In *Proceedings of the 12th Web for All Conference* (W4A '15). ACM, New York, NY, USA, Article 8, 4 pages. DOI=http://dx.doi.org/10.1145/2745555.2746650

[13] Kovacs, G. and Miller, R. C. 2014. Smart subtitles for vocabulary learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 853-862. DOI=http://dx.doi.org/10.1145/2556288.2557256

[14] Kruger, J. L., Hefer, E., and Matthew, G. 2013. Measuring the impact of subtitles on cognitive load: eye tracking and dynamic audiovisual texts. In *Proceedings of the 2013 Conference on Eye Tracking South Africa* (ETSA '13). ACM, New York, NY, USA, 62-66. DOI=http://dx.doi.org/10.1145/2509315.2509331

[15] Kushalnagar, R. S., Lasecki, W. S., and Bigham, J. P. 2013. Captions versus transcripts for online video content. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility* (W4A '13). ACM, New York, NY, USA, Article 32, 4 pages. DOI=http://dx.doi.org/10.1145/2461121.2461142

[16] Liu, X. and Wang, W. Robustly extracting captions in videos based on stroke-like edges and spatio-temporal analysis. *IEEE Transactions on Multimedia*. 14, 2 (Apr. 2012), 482–489. DOI=http://dx.doi.org/10.1109/TMM.2011.2177646

[17] Rooney, K. 2014. The Impact of Keyword Caption Ratio on Foreign Language Listening Comprehension. *International Journal of Computer-Assisted Language Learning and Teaching*. 4, 2 (Apr. 2014), 11-28. DOI=http://dx.doi.org/10.4018/ijcallt.2014040102

[18] Shimogori, N., Ikeda, T. and Tsuboi, S. 2010. Automatically generated captions: will they help non-native speakers communicate in english?. In *Proceedings of the 3rd international conference on Intercultural collaboration* (ICIC '10). ACM, New York, NY, USA, 79-86. DOI=http://dx.doi.org/10.1145/1841853.1841865

[19] Tiedemann, J. 2008. Synchronizing translated movie subtitles. In *Proceedings of the 6th International Conference on Language Resources and Evaluation* (LREC '08), 5 pages.

[20] Villa Real, L. C., Guimarães, R. L., and Avegliano, P. 2015. Dynamic Adjustment of Subtitles Using Audio Fingerprints. In *Proceedings of the 23rd ACM international conference on Multimedia* (MM '15). ACM, New York, NY, USA, 975-978. DOI=http://dx.doi.org/10.1145/2733373.2806378

[21] Wang, F., Nagano, H., Kashino, K., and Igarashi, T. 2015. Visualizing video sounds with sound word animation. In *Proceedings of the IEEE International Conference on Multimedia and Expo* (ICME '15). 1-6. DOI=http://dx.doi.org/10.1109/ICME.2015.7177422