# Automatic Creation and Timing of Subtitles based on Sound Analysis

BY
Ignatios Charalampidis

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Arts in the Science of Translation in the Department of Foreign Languages, Translation and Interpreting of the Ionian University on Corfu, 2021

Corfu, Greece

Defense Committee: Dr. Vilelmini Sosoni, Dr. Maria Tsigou, Dr. Katia-Lida Kermanidou

**Abstract**

In this paper I explain the inner workings of IgniSub, a piece of software that attempts to generate subtitles given any audiovisual material. The software's purpose is to create and sync subtitles based on various methods of sound analysis, namely voice activity detection, speaker diarization and parameter tweaking. Firstly, a machine learning algorithm is used to find regions of speech. Then, another algorithm further segments speech regions according to different speakers. Finally, subtitles are generated based on a scoring system which takes all factors into account.

**Περίληψη**

Στην παρούσα εργασία επιχειρώ να περιγράψω τη λειτουργία του IgniSub, ενός λογισμικού που έχει ως στόχο την αυτόματη δημιουργία υποτίτλων δεδομένου οποιουδήποτε οπτικοακουστικού υλικού. Ο σκοπός του λογισμικού είναι η παραγωγή και ο χρονισμός υποτίτλων βάσει διαφόρων μεθόδων ανάλυσης ήχου, ήτοι μέσω εντοπισμού ομιλίας, διαχωρισμού ομιλητών και παραμετροποίησης. Αρχικά ένας αλγόριθμος μηχανικής μάθησης εντοπίζει τα σημεία που περιέχουν ομιλία. Κατόπιν, ένας άλλος αλγόριθμος κάνει διαχωρισμό ομιλητών και επιπλέον κατάτμηση των περιοχών ομιλίας. Τέλος, δημιουργούνται υπότιτλοι σύμφωνα με ένα σύστημα αξιολόγησης που αναλύει όλα τα δεδομένα.

Table of Contents

1.1 Introduction....4
1.2 Scope of the project....4
1.3 Existing technologies combined for a new purpose....5
2.1 Subtitling limitations, conventions and common practices....5
2.2 Subtitling, semantics, pragmatics and the limitations of artificial intelligence....7
2.3 The challenge of automatic subtitle generation....8
3 Approach and methods....9
3.1.1 Voice Activity Detection....9
3.1.2 VAD method used in this project....11
3.2 Speaker diarization for subtitle boundary estimation....11
3.2.2 The speaker diarization method used in this project....12
3.3 Subtitle finalization - Tying it all together....13
4. Optimization, Debugging and Future Work....14
5. References....15

List of Abbreviations

| | |
|---|---|
| ST | Source text |
| TT | Target Text |
| SL | Source Language |
| TL | Source Language |
| CNN | Convolutional Neural Network |
| DNN | Deep Neural Network |
| VAD | Voice Activity Detection |
| STD | Standard deviation |
| FFT | Fast Fourier Transform |
| NLP | Natural Language Processing |
| HMM | Hidden Markov Model |
| STT | Speech to Text |

## 1.1 Introduction

In this paper the process of creating IgniSub is laid out and analyzed. IgniSub is a standalone piece of software that caters to the creation of subtitles for any audiovisual material. Unlike regular subtitling programs, IgniSub has the added benefit of automatically generating and timing subtitles. To that end machine learning algorithms or used, and in particular neural networks, which are state of the art when it comes to machine learning. Unlike existing systems that rely on external, online services in order to carry out similar tasks, IgniSub is driven by the need to have a standalone offline application that functions independently, without the necessity to use online paid and premium resources and services, such as those provided by Google.

The present paper is separated into two parts, namely the theoretical background concerning subtitling and the technical details of creating the software itself. More specifically, an overview of all the design decisions is given, as well as the reasoning behind those decisions based on the theory and common practices of subtitling.

## 1.2 Scope of the project

In order to define what this project is about I think that it would be best to first state what it is not: 1) IgniSub is not an end-to-end program that generates subtitles fully automatically. Rather, it is meant to be used as a tool to aid the subtitling process. Therefore, many tasks still have to be performed manually, though the workload of traditional subtitling pipelines should be reduced significantly. At the time of writing IgniSub is capable of only creating and timing closed captions based on the audio material at hand. Speech recognition, though implemented, is an experimental feature and as such it lies

outside the scope of this paper. In the future it should be possible for the program to handle both transcription of audio and timing of subtitles with ease, but the focus of the present work is almost exclusively on generation and timing of subtitles. 2) IgniSub is not a machine translator, like popular systems (e.g. Google Tanslate, Systran etc). At the time of writing no such functionality is implemented, even though the addition of such capabilities in the future is under consideration. 3) Finally, IgniSub is not a computer aided translation (CAT) tool, meaning that it does not provide features such as those offered by professional CAT software (e.g. Trados, MemoQ, DejaVu etc.). In other words, IgniSub is not a translation memory tool nor does it make use of term databases. One can think of it as a piece of software akin to Subtitle Edit and AegiSub.

It is to be mentioned that in the present paper the focus is mainly on the subtitling aspect and less so on the technical one. Of course, a broad overview of the technology used is provided, but without delving deeper into the mathematical intricacies of the algorithms used. After all this is an attempt to examine how new technologies could be deployed in the field of subtitling and what limitations and/or idiosyncrasies could be overcome using them. A detailed account of the inner workings of the algorithms is out of scope and is not to be expected.

### 1.3 Existing technologies combined for a new purpose

Timing captions is undoubtedly one of the more tedious tasks in subtitling. Various limitations and rules apply, which render subtitle synchronization a time-consuming, and sometimes even challenging, endeavor. For that reason I feel it is worth investing in technologies that lift the weight off the subtitler's shoulder and automate the process to the largest extent possible.

### 2.1 Subtitling limitations, conventions and common practices

In order to devise a tool that automatically generates subtitles, one would have to consider the idiosyncratic nature of subtitling as a form of translation. Subtitling is not exactly translation per se nor is it a free form of paraphrasing. It lies somewhere in between, for it combines the merits (and limitations) of both worlds. In an effort to systematically study audiovisual translation, Henrik Gottlieb delineates those limitations in [1]. As Gottlieb explains, limitations are quantitative and qualitative in nature. Temporal and spatial constraints render subtitling a challenging task, for one thing because captions should not exceed a certain character limit and for another because captions are meant to appear on screen only for a certain duration. Due to these constraints subtitles need to adhere to strict

rules regarding formatting and timing. This necessitates syntactic, stylistic and vocabulary shifts in order for the TT to conform to said contraints. Therefore, meaning and content in the TT are also bound to suffer some quality loss, as they have to be condensed and altered in the TL more often than not. What is more, established subtitling conventions make it so that professional translators cannot deviate from common practices. Translators who subtitle movies and series have to constantly strike a balance between loss of meaning and subtitle length, effectively trading accuracy for readability and vice versa. It is often the case that the meaning of the ST cannot be rendered exactly in the target language only because a subtitle's length exceeds the allowed character limit by one single character. Translators have to abide by the commissioner's limitations and rules regarding subtitling, many of which are immensely restricting.

Researcher Fotios Karamitroglou examines common practices as regards subtitling in Europe and especially in Greece. In [2] Karamitroglou lists a series of common guidelines in subtitling. These may vary across different commissioners and content providers, but they serve as a reliable rule of thumb for most professional subtitling tasks. As far as temporal and spatial restrictions are concerned, in [2] it is stated that the average number of words the audience can read while simultaneously watching the audiovisual material amounts to roughly two to three words per second. That translates to about 35 characters per second, which of course varies across different age groups and education backgrounds. [3] seems to agree with this limit, which concludes that the human brain needs five to six seconds to process 70 characters (two lines in the subtitle), while at the same time processing the rest of the stimuli, namely the image and sound. Therefore, it is not advisable to add 70 characters when the subtitle lasts for only one second, because there simply is not enough time for the audience to read it. In [2] It is also mentioned that no subtitle should appear for anything less than one second.

Cintas and Remael in [4] seem to agree, asserting that a subtitle should contain no more than two lines with 37 ± characters per line and should not exceed the duration of six seconds. To complicate things further, a closed caption's duration should be proportional to the information conveyed during said caption's appearance. More words spoken imply a longer caption duration. Lastly, a quick succession of subtitles is to be avoided, as Karamitroglou asserts in [2], because viewers need some time to process the change from one subtitle to the next. Therefore, an amount of time is preferable between subtitles. That amount of time is empirically found to be around 0.25 seconds, but there is no consensus.

Different content providers impose their own limitations and rules when it comes to subtitling. Netflix provide guidelines, according to which the in-time of a subtitle should be within 1-2 frames from the beginning of an utterance and out-time can be extended for up to half a second after the utterance has

finished [5]. Where there are pauses between utterances that are less than half a second long, the subtitles should be chained, leaving 2 frames of space between each subtitle. Moreover, where there is a shot change, in-time should be adjusted accordingly so that the subtitle starts at the frame of the shot change. The same goes for dialogue, where shot changes should coincide with speaker turns. Consecutive subtitles should have a minimum gap of 2 frames between them and if a subtitle has already been extended for half a second, merging and resegmenting should be considered as options instead of further extending said subtitle.

In order for an automated system to adhere to those rules, there needs to be a way to determine how much information is conveyed, which would prove useful in allocating the right amount of time for each caption, as well as in splitting captions at the right point. However, this proves to be a particularly difficult task, since it requires a means of linguistic content extraction and a way of detecting scene/shot transitions.

### 2.2 Subtitling, semantics, pragmatics and the limitations of artificial intelligence

Translation is a science as much as it is an art, and subtitling is no exception. Knowing how to transfer meaning from one language to another is already a baffling task, let alone between different media and/or modalities. As if that was not enough, teaching a machine to do the same thing poses a greater challenge, because even state of the art algorithms are still not able to fully tackle that problem. Our best systems produce impressive translations, but that is still a far cry from an end-to-end system that handles such complicated tasks as translation, summarization and context retrieval all at once. The reason for that seems to lie in fact that machines, though capable of completing a plethora of tasks previously inconceivable, still lack creative thinking and context awareness. In other words, machines lack semantic and pragmatic knowledge of the world and the text they are given. The implications of these restrictions are obvious when it comes to translation per se, but they are examined here for another, less conspicuous reason: since subtitles reflect what is being said on an utterance level, a deep understanding of grammar, syntax and semantics is required for a successful segmentation of subtitles.

Professional translators have some years of formal training and a good deal of experience when it comes to translating, but even an untrained layman can effortlessly grasp fundamental syntactic and semantic relations and can relate a text to the world around them. Failure to do so produces translation errors, like in the case of polysemy, where wrong interpretation might result in peculiar and sometimes amusing translations. This skill is paramount to subtitling as well because subtitles are supposed to be logically structured. Ideally, each caption should begin when a new utterance is spoken and span across the duration of the utterance until the next one appears. Being able to carry out this seemingly trivial

task requires knowledge which is intrinsic to humans, but not machines, namely knowledge of syntax, semantics and pragmatics.

Roman Jacobson attempted to provide a systematic study of language, one that many theorists called Structuralism, according to which meaning is constructed by the relations various components have in a given language. In [6] Jakobson maintains that words lead to understanding only when the person perceiving them has an even faint acquaintance with the meaning they allude to. Apart from the denotative meaning of words, however, understanding develops even from extra-textual information. Mona Baker in [7] defines pragmatics as the study of how meaning is constructed by the speakers themselves and not the text's underlying structure. In other words, pragmatics examines the meaning of words or phrases as it emerges from what readers / speakers already know, expect or can infer, that is, what is often implied but not explicitly expressed.

Finally, on a more rudimentary level meaning is constructed by the grammatical and syntactical relations between the elements in a language. In [8] Noam Chomky purports that certain grammatical-syntactic structures govern all natural languages and are an inherent, innate part of human understanding, something which he calls Universal Grammar.

In NLP such understanding of language is crucial, but its implementation is costly and not straightforward. In subtitling in particular building a system that properly segments and synchronizes captions based on linguistic content is a taxing task, which requires a great amount of research. OpenAi's GPT-3, a DNN [28], shows promising results in language comprehension, but the sheer amount of memory it needs to operate and its immense requirements on disk space make it unsuitable for systems such as IgniSub, which is aimed at translators.

### 2.3 The challenge of automatic subtitle generation

From the above it becomes apparent that automatically synchronizing subtitles is no easy task. For a start, subtitles need to follow the syntactic and semantic flow of the utterances spoken. As of the time of writing, there is no end-to-end solution that effectively takes care of segmentation in speech. That would require a system that can detect suprasegmental features, such as intonation, and even meaning so as to determine when utterances begin and when they end. To further illustrate this point, in spontaneous, everyday speech false starts and fillers can be found in abundance. That makes it almost impossible for a computer to determine the boundaries of utterances without some form of content recognition. Of course, there are STT systems that are fairly good at transcribing speech to text, but even they have their limitations; although capable of accurately converting speech to text, such systems

tend to produce a continuous sequence of words without punctuation, thus proving themselves unhelpful in utterance boundary detection. When combined with automatic punctuation technologies, they show promising results, but transcription needs to be practically flawless for such systems to work properly.

To make matters worse, even if reliable content recognition systems existed, at the moment it is rather unreasonable to expect STT (or any audio-based technology for that matter) to perform flawlessly under any conditions. In most cases speech is muffled, overlaid with noise or music or outright inaudible. Besides, STT is not ripe enough for the majority of languages, which makes it useful for only a handful of languages.

Furthermore, utterance boundary detection and linguistic content are not the only factors that come into play in subtitling. Speech rate, the existence of dialogue or lack thereof and informativity are all factors that a subtitler takes into consideration. Subtitling is not like traditional translation and often requires different strategies, such as condensation, deletion and paraphrase [1]. Depending on the content and on the genre of the audiovisual material at hand, the subtitler must make the right decision when it comes to subtitle creation. Some utterances can easily be paraphrased or condensed, thus requiring a shorter subtitle, while others cannot, therefore requiring more than one subtitle and more careful placement. Across various genres subtitling style changes too; In situations where language is scripted (e.g. TV show, movie, rehearsed speech) subtitles are expected to follow utterance boundaries more closely, since such texts are more coherent and less prone to errors and false starts. In movies and series, one utterance expresses one idea most of the time, whereas in interviews and unscripted situations speech is more chaotic with redundant repetitions, overlapping speakers and digressions. In such cases linguistic content awareness is not enough for proper subtitling, which makes the task even harder for machine learning algorithms to execute.

## 3 Approach and methods

Without a proper way of detecting utterance boundaries and recognizing linguistic or extra-linguistic context, I opted for a simpler approach. In order to simplify the task of automatic subtitle generation, I chose to break it down into three different stages: 1) Voice Activity Detection, 2) Speaker Recognition and 3) Parameter optimization. If each stage is executed adequately, we should, in theory at least, end up with decent subtitles. In the following segments the process is outlined for each stage.

### 3.1.1 Voice Activity Detection

Voice Activity Detection (VAD) is the cornerstone of this project. It is defined as the task of detecting human voice in a given snippet of audio. That entails excluding anything that does not qualify as human speech and returning only the parts that contain meaningful speech information. That poses the challenge of not only detecting non-speech, but also filtering out human sounds that are not meaningful, hence not used in subtitles. Such sounds include, but are not limited to, shouts, exclamations and fillers such as “er” and “uhm”. VAD is essential for such an undertaking as automatic subtitling.

Various methods have being explored with varying levels of success. The most bare-bones implementation involves checking the volume of a given snippet, as done in [9] and [10]. Modified low energy ratio is a method whereby audio is split into frames of 20-25 ms. Then, for windows of 250-1000ms we count how many frames are above a certain volume threshold, marking them as voiced. If the ratio of voiced and unvoiced frames is within a specified range, then the window is marked as containing speech. In [11] the goal is again to discriminate between speech and music and to that end RMS and zero-crossings are used. Segmentation is carried out based on mean signal amplitude distribution and classification is performed using frequency features.

The intuition behind these methods is that speech consists of voiced and silent phonemes, thus creating a pattern of low-volume and high-volume frames of audio. If a given window contains music, then the volume across frames is more evenly distributed. Finally, if said window contains background noise, then the pattern would consist of low-volume frames. The issue arises when the audio in question contains speech overlaid with music or background noise. Then it is difficult to discern where speech begins and ends. What is more, there are music patterns that seem to follow the pattern of speech, with a quick succession of low and high-volume frames. Unfortunately, most series and movies contain such sequences of speech mixed with music and/or noise. Good news is that such occurrences are scarce.

A more robust approach to VAD is the use of neural networks. Notwithstanding the extra processing power and training time, neural nets seem to provide more consistent results more frequently. This method’s strength lies in the fact that neural nets excel at “picking” the features that are most relevant for each task. Unlike other, simpler methods, neural nets are not bound by a certain set of features, but rather find the features, or combinations thereof, that are most useful.

In [12] a neural network was trained with what the authors call Multi-resolution stacking (MRS), an ensemble of classifiers that extract contextual information from sound. [13], [14] and [15] all utilize neural networks to perform VAD based on acoustic features. A more novel approach can be found in [16] where instead of frequencies, the images of spectrograms are used as features to a CNN, which

learns to discriminate between music and speech.

### 3.1.2 VAD method used in this project

IgniSub utilizes two methods of VAD, one being a DNN and the other being a simpler volume-based method. For the former I trained a neural net on a custom dataset curated for the purposes of this research. The dataset contains audio from various sources marked for speech, music and noise. The problem of VAD in this case is framed as a classification problem, much like in [12]. First the Fast Fourier Transfrom is performed on the signal so as to acquire the frequencies of the sound. Then the MFCCs are extracted, which are in turn used as features for the neural network to learn from. A sliding window splits the audio into fixed-size windows, each of which is run through a CNN with the purpose of being marked as either containing human speech or not. Simply put, each segment of audio is classified as speech or non-speech based on its surrounding context.

This takes care of large segments of speech, but it often fails to detect pauses shorter than 250ms. In other words, short gaps between speech segments are considered as speech. For the purposes of this paper, shorter gaps may need to be recognized in order for the program to function properly. Therefore another method is used for that. Much like [9] and [10], the segments of speech detected by the neural net are rescanned by an algorithm that looks for frames of sound (typically 25ms each) whose energy is lower than a given percentage of said segment's average energy. In a nutshell, if a series of consecutive frames is much lower in volume than the average of the speech segment, then that sequence is considered to be a gap.

## 3.2 Speaker diarization for subtitle boundary estimation

Speaker diarization is defined as the task of determining who spoke when. This is useful for subtitle generation in cases where there are long speech segments with no pauses or silences in between, thereby calling for a way to delineate utterance boundaries. The idea is that a subtitle should begin at the onset of a speaker's utterance and finish either when another speaker cuts in, if the two utterances are sufficiently long, or when the second speaker finishes their utterance, if the two utterances are short enough to be merged into one subtitle, in which case a subtitle containing a dialogue is generated.

Traditional approaches to this task have revolved around the use of i-vectors, GMMs and clustering methods. In [17], [18] and [19] such methods are used for speaker diarization. Simply put, some acoustic features are extracted, a statistical model generates a fixed-size vector for every segment (called an i-vector) and then a machine learning algorithm compares those vectors and determines whether they belong to the same or different speakers.

More recent and more robust techniques involve the use of d-vectors or x-vectors, namely fixed length vectors generated by DNNs. Those methods can be supervised, meaning that the algorithm is trained to do speaker diarization by looking at examples, or unsupervised, in which case an algorithm sorts the generated d-vectors into homogeneous groups or clusters with each one typically containing utterances from the same speaker. To my knowledge, the state of the art in supervised speaker diarization is [20], where a RNN encodes utterances of speakers and then uses statistical parameters to determine whether a new utterance belongs to a previously seen speaker or a new one. Though accurate, this system is growing exponentially slower with each second of audio it has to decode and is therefore not recommended for this project whose main goal is fast implementation on low-end hardware.
Other papers mostly refer to unsupervised methods, such as those described in [21] and [22] where d-vectors are clustered using a GMM. In [23] and [24] d-vectors are coupled with a Kmeans algorithm, which attempts to cluster the produced embeddings (d-vectors) of a DNN into separate speakers.

### 3.2.2 The speaker diarization method used in this project

The ultimate purpose of the aforementioned methods is to estimate the points of speaker change in a given clip of audio. To that end I used a method similar to the ones described in [23, 24, 25] because I found it to be the fastest, albeit less accurate. More specifically, audio is split into segments of one second. Then for intervals of half a second [26] is used to extract the d-vectors. That effectively leads to overlapping segments, each of which has its own d-vector representation. [26] is a system that is based on [27], a speaker verification method, and it was trained on a huge dataset of speakers with the end goal of producing a vector representation of a speaker's identity given only a small audio sample. The idea is that vectors produced by different speakers should have a bigger euclidean distance between them than those produced by the same speaker.
Then, an algorithm similar to [23] and [24] is used in order to cluster those vectors in groups representing different speakers. I chose to use a simple Kmeans algorithm with a default of 20 components because it is fast. However, there is a tradeoff; Unsupervised clustering methods, such as GMMs and Kmeans are stochastic in nature. In lay-people's terms, this means that there is a component of randomness involved, In other words, these algorithms return results which can be drastically different across sessions running on the same data. This has to do with the random order in which these algorithms examine the given samples. More accurate machine learning algorithms tend to produce stable and more consistent results at the cost of extra time and more computing power. However, since IgniSub is meant to run on low-end hardware, such resources are limited and we cannot afford the extra time. Therefore, finding the right algorithms to use is a constant trade-off between performance and

speed.

Finally, the results are rectified by going through what I call the *distillation* process. Kmeans splits the segments into 20 homogeneous clusters. Once clusters are determined, the median of the clusters' d-vectors is computed. The result is a 256-dimension vector for each cluster, which acts as the profile of the whole cluster. These profiles are compared to one another using a cosine similarity metric and if the result for any of the profiles is found to be larger than a given threshold (typically in the range of 0.75-0.8), then the respective clusters are merged into one. This effectively reduces the number of clusters to fewer than 20. Then the members of each cluster are reexamined to check whether they truly fit with the average of the cluster. If they do not, they are labeled as "outcasts", they are removed from the cluster and in later iterations of the distillation process they are compared with all the clusters' profiles again using the same cosine similarity metric. If the "outcasts' " similarity with another cluster is high enough, then they are reassigned to that cluster, otherwise they form an "anonymous" cluster on their own. After this process is completed the user can manually merge clusters for maximum speaker diarization accuracy.

**3.3 Subtitle finalization - Tying it all together**

So far the individual components of this paper have been analyzed. In this section an explanation is provided as to how those building blocks combine in order to produce the final result. Using VAD loosely defined segments of human speech are found. However, these segments alone do not constitute the finalized subtitles because a single segment of continuous speech may contain utterances from different speakers. Additionally, in the case of dialogue where a quick exchange of utterances from different speakers takes place, more than one caption should be created, since captions should not stay on screen for longer than six seconds, as mentioned in 2.1. Another issue is that VAD segments are often too short or too long to become subtitles, which necessitates their merging with neighboring subtitles or their breakdown into smaller pieces.

Therefore, for every VAD segment a decision needs to be made as to whether said segment will be merged with the neighboring segments on either side or stay as is. Similarly, VAD segments that are longer than 6 seconds should be split into more segments, which in turn become candidates for merging themselves.

In order to avoid verbose code and multiple conditional statements, a formula was devised which assigns a score to each VAD segment. This score can be either positive or negative, the former signifying a tendency to merge with the previous VAD segment and the latter denoting a tendency to merge with the next segment. This is essentially an elaborate game of "tug of war", where each

segment of speech is “attracted” by its neighbors so as to merge with them. A merge occurs when the score of the segment in the middle is sufficiently big in either the positive or negative direction. If the absolute value of the score is lower than a given (and customizable) threshold, then the speech segment in question remains stationary, thereby not merging with any of its neighbors.

The formula that determines each segment’s score is as follows:

$$Score = \left(-\left(c/S_{i-1}\right)-\left(q/D_i\right)-W_{i-1}+\left(c/S_i\right)+\left(q/D_{i+1}\right)+W_{i+1}\right)*f$$

where S is the silence before and after each speech segment in question, c and q are given factors, D is the total length that the speech segment in question would be if it was merged with the previous or next segment, and W is a factor that fluctuates depending on whether the segments adjacent to the segment in question belong to the same speaker or not. Finally, f is a factor determined by the total length of the speech segment in question.

What this formula effectively does, is it favors merging between speech segments that are either very close to one another (as indicated by the silence between them) or speech segments that are too short to stand alone. The intuition behind this logic is that speakers are more likely to utter a phrase in one go instead of breaking it up in smaller, distant pieces. This is especially the case with scripted language, where an utterance is said in one breath, thus leaving no pauses or silences between words.

The influence of speaker change can also be controlled, thereby favoring merging among segments of the same speaker or merging among segments that belong to different speakers. It also acts as an extra breaking point when speech segments are too long.

Segments of speech that are separated by a time gap smaller than 0.2 seconds are good merge candidates. This means that those segments could be merged if all conditions are met. These conditions are that the merged subtitles not exceed the limit of six seconds and that the amount of information conveyed by the candidates is not too high to be contained in too small a time frame. Also, if multiple candidates exist in a row, merging priority is given to those that form dialogues and those that constitute independent semantic units (as currently judged by STD metrics and speaker change indicators).

Finally, formatting takes all of the above into account. If a change of speakers, and hence a dialogue, is detected, then the generated subtitle will contain two lines starting with dashes so as to signify to the translator that a dialogue is the recommended course of action.

## 4. Optimization, Debugging and Future Work

IgniSub is by no means a finished product. Future work involves fine-tuning the system's hyperparameters for optimal results, as well as exploring other methods and algorithms for a more efficient and accurate implementation. Performance optimization is also in the pipeline, as the program currently runs less than optimally. Since this project is a proof of concept, future iterations shall consume less memory, be less hardware-intensive and more performant, harnessing the capabilities of multithreading and delivering more accurate results in a shorter amount of time. Refactoring of the codebase, as well as debugging and polishing of the user interface are to be completed.

Furthermore, the capabilities of the software will be extended with the addition of visual analysis methods that take visual content into account when generating subtitles. Such methods could include face and facial expression recognition, which could prove useful in the process of speaker diarization and scene detection for more accurate placement of subtitles. Support for Subtitles for the deaf and Hard of Hearing (SDH) is also underway, with the groundwork already set.

Finally, since the task of creating subtitles requires comprehension of the content of the audiovisual material at hand, the implementation of suprasegmental analysis and semantic parsing would be a valuable addition to the existing project. For instance, more accuracy and efficiency could be achieved by utilizing machine learning algorithms that pertain to automatic text summarization, a tricky yet invaluable task when it comes to subtitling. Better results could also be yielded if there was a reliable way to detect changes in intonation, mood and speech rate, all of which are functions that were not added due to time constraints.

## 5. References


[1] Gottlieb, Henrik. "Subtitling-a new university discipline". In *Teaching translation and interpreting*. John Benjamins, 1992. 161.

[2] Karamitroglou, Fotios. "A proposed set of subtitling standards in Europe".*Translation journal* 2, no.2 (1998): 1–15.

[3] Hanson, Göte. *Läsning av text i tv*.Sveriges Radio, Publik-och Programforskningsavdelningen, 1974.

[4] Cintas, Jorge Díaz, and Aline, Remael. *Audiovisual translation: subtitling*.Routledge, 2014.

[5] “Timed Text Style Guide: Subtitle Timing Guidelines.” Netflix. Last modified July 09, 2021. https://partnerhelp.netflixstudios.com/hc/en-us/articles/360051554394-Timed-Text-Style-Guide-Subtitle-Timing-Guidelines

[6] Jakobson, Roman. *On linguistic aspects of translation*.Harvard University Press, 2013.

[7] Baker, Mona. *In other words: A coursebook on translation*.Routledge, 2018.

[8] Chomsky, Noam. *Aspects of the Theory of Syntax*. Vol. 11. MIT press, 2014.

[9] Ericsson, Lars. *Automatic speech/music discrimination in audio files*.Citeseer, 2010.

[10] Wang, WQ, W, Gao, and DW, Ying. "A fast and robust speech/music discrimination approach." . In *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint* (pp. 1325–1329).2003.

[11] Panagiotakis, Costas, and Georgios, Tziritas. "A speech/music discriminator based on RMS and zero-crossings".*IEEE Transactions on multimedia* 7, no.1 (2005): 155–166.

[12] Zhang, Xiao-Lei, and DeLiang, Wang. "Boosting contextual information for deep neural network based voice activity detection".*IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24, no.2 (2015): 252–264.

[13] Wang, Linlin, Chao, Zhang, Philip C, Woodland, Mark JF, Gales, Panagiota, Karanasou, Pierre, Lanchantin, Xunying, Liu, and Yanmin, Qian. "Improved DNN-based segmentation for multi-genre broadcast audio." . In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5700–5704).2016.

[14] Eyben, Florian, Felix, Weninger, Stefano, Squartini, and Björn, Schuller. "Real-life voice activity detection with lstm recurrent neural networks and an application to hollywood movies." . In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 483–487).2013.

[15] Hughes, Thad, and Keir, Mierle. "Recurrent neural networks for voice activity detection." . In

*2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 7378–7382).2013.

[16] Papakostas, Michalis, and Theodoros, Giannakopoulos. "Speech-music discrimination using deep visual feature extractors".*Expert Systems with Applications* 114 (2018): 334–344.

[17] Kenny, Patrick. "Bayesian speaker verification with heavy-tailed priors.." . In *Odyssey*.2010.

[18] Brümmer, Niko, and Edward, De Villiers. "The speaker partitioning problem.." . In *Odyssey* (pp. 34).2010.

[19] Garcia-Romero, Daniel, and Carol Y, Espy-Wilson. "Analysis of i-vector length normalization in speaker recognition systems." . In *Twelfth annual conference of the international speech communication association*.2011.

[20] Zhang, Aonan, Quan, Wang, Zhenyao, Zhu, John, Paisley, and Chong, Wang. "Fully supervised speaker diarization." . In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6301–6305).2019.

[21] Zajíc, Zbynek, Marek, Hrúz, and Ludek, Müller. "Speaker Diarization Using Convolutional Neural Network for Statistics Accumulation Refinement.." . In *INTERSPEECH* (pp. 3562–3566).2017.

[22] Shum, Stephen H, Najim, Dehak, Reda, Dehak, and James R, Glass. "Unsupervised methods for speaker diarization: An integrated and iterative approach".*IEEE Transactions on Audio, Speech, and Language Processing* 21, no.10 (2013): 2015–2028.

[23] Wang, Quan, Carlton, Downey, Li, Wan, Philip Andrew, Mansfield, and Ignacio Lopz, Moreno. "Speaker diarization with LSTM." . In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5239–5243).2018.

[24] Dimitriadis, Dimitrios, and Petr, Fousek. "Developing On-Line Speaker Diarization System.." . In *INTERSPEECH* (pp. 2739–2743).2017.

[25] Ning, Huazhong, Ming, Liu, Hao, Tang, and Thomas S, Huang. "A spectral clustering approach to speaker diarization." . In *Ninth International Conference on Spoken Language Processing*.2006.

[26] “Resemblyzer.” Github. Last modified June 12, 2020.

[27] Wan, Li, Quan, Wang, Alan, Papir, and Ignacio Lopez, Moreno. "Generalized end-to-end loss for speaker verification." . In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4879–4883).2018.

[28] Brown, Tom B, Benjamin, Mann, Nick, Ryder, Melanie, Subbiah, Jared, Kaplan, Prafulla, Dhariwal, Arvind, Neelakantan, Pranav, Shyam, Girish, Sastry, Amanda, Askell, and others. "Language models are few-shot learners".*arXiv preprint arXiv:2005.14165* (2020).