Universidad Nacional
de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Speak2Subs: Evaluating State-of-the-Art Speech Recognition Models and Compliant Subtitle Generation

Julio Antonio Fresneda García

Director/a: Jorge Pérez Martín
Co-director/a: Álvaro Rodrigo Yuste

Trabajo de Fin de Máster

Máster Universitario
en Ingeniería y Ciencias de Datos

Febrero 2024

Acknowledgments

To my amazing girlfriend and family who've braved my academic hibernation throughout this project.

To Dire Straits.

Abstract

With recent advances in largue language models, the evolution of speech-to-text tasks has been exponential. While state-of-the-art automatic speech recognition (ASR) models have taken a big step in speech transcription, creating quality subtitles still requires human intervention.

This project has two main aspects: evaluating cutting-edge ASR models for speech-to-text, and developing a package that uses this ASR models to generate high-quality and compliant subtitles.

ASR models do not inherently provide results suitable for subtitles. Therefore, one of the primary objectives of this package is to utilize and enhance the output generated by ASR models to create subtitles of a quality that requires minimal human modification. This enhancement is necessary because ASR models alone are incapable of producing subtitles that meet the required standards of quality.

Speak2Subs has achieved this goal, being a tool that produces high-quality subtitles with minimal human interaction.

Keywords: ASR, LLM, Speech-To-Text, Subtitle.

Glossary

ASR Automatic Speech Recoginition is a technology that converts spoken language into written text, enabling machines to transcribe and understand human speech. It plays a crucial role in applications like voice assistants, transcription services, and subtitling..

Container A self-contained, portable unit of software that encapsulates applications along with their dependencies, ensuring consistent and isolated execution across diverse computing environments..

LLM Large Language Models (LLMs) are artificial intelligence models, characterized by their extensive scale and complexity. These models, trained on vast amounts of data, excel in understanding, generating, and processing human-like language. An example is ChatGPT..

Python package A collection of Python modules and related resources organized in a directory structure file, allowing for the encapsulation, reuse, and distribution of code and functionalities. Packages facilitate modular programming and enable namespace management within Python projects..

UNE Spanish Association for Standardization (Asociación Española de Normalización). UNE develops and publishes technical standards in various fields to ensure consistency and quality in products, services, and processes across Spain. These standards cover areas such as industry, technology, safety, and environmental practices..

Contents

List of Figures

Table index

Chapter 1

Introduction

1.1 Brief context

The art of subtitle generation, like many other disciplines, has undergone significant evolution in recent years. While tools have existed for quite some time to ease this task, it has always demanded substantial human supervision and involvement. Technological progress over the years has been noteworthy, and with the emergence of advanced language models like ChatGPT [1], LLama [17], Whisper [14], and others, the field has experienced a remarkable leap forward. These models now demonstrate exceptional accuracy in speech-to-text tasks, an accuracy that was once unimaginable.

However, the creation of quality subtitles needs more than just accurate transcription. Although we can obtain a precisely transcribed text with timestamps for each word, the art of structuring these words into coherent subtitles still demands human intervention, presenting a task that remains somewhat difficult to do.

This master's thesis has two integral aspects. Firstly, it involves the development of a robust package using the most advanced automatic speech recognition (ASR) models. Leveraging the capabilities of the ASRs, this tool aims to generate high-quality subtitles that adhere to specific conditions and checks, ensuring both proper readability and seamless integration with the original source material. Secondly, this thesis has a comprehensive evaluation and comparison of the current state-of-the-art ASR models, specifically those available as open-source, specialized transcribing speech to text.

To carry out this evaluation, we have achieved a package that minimizes human intervention to the greatest extent possible in the intricate task of subtitling. By bridging the gap between cutting-edge ASR technology and the artistry of subtitling, this project seeks to push the boundaries of automation and enhance the accessibility and efficiency of subtitle creation.

[1] `https://chat.openai.com`

1.2 Motivation

The motivation to make this idea of project into a real thing comes from two sources. There is an inherent motivation in knowing that we can help to evolve this field and contribute with something at the cutting edge of the art of generate subtitles.

This process demands a level of linguistic and contextual comprehension that traditionally aligns with human capabilities. The intricacies embedded in linguistic nuances, cultural references, and contextual awareness have, until now, posed a formidable barrier to automated systems. The human interaction seems inevitable in this task, but with the upraisal of some news technologies, we have the oportunity to work in reduce this interaction to the minimum. With this project, we can consider a humble step has been taken in this direction.

There is a personal motivation too. Within the academic realm, specifically at the Universidad Nacional de Educación a Distancia (UNED [2]), there exists a pressing need for an efficient tool that seamlessly generates subtitles with minimal human intervention. However, the current process of transmuting spoken content into subtitles poses a bottleneck that slows down the institution's commitment to providing accessible and inclusive educational resources. The personal motivation to address this challenge lies in the quest for a tool that would not only enhance the efficiency of content delivery but also contribute significantly to reducing the workload on human transcribers.

In essence, the dual motivation to address the global challenge of transforming transcriptions into subtitles and the specific needs of UNED, and the deep measure and evaluation of state-of-the-art ASR models, converges in the quest for a out-of-the-box package that can help in the task of making subtitles with excellent quality.

1.3 Proposal and objectives

The principal aim of the project is to enhance the quality of automatic subtitling. To achieve this, we have developed a framework, a Python package, capable of generating high-quality subtitles. These subtitles are designed to adhere to specific regulations while requiring minimal human editing. Supporting this main goal, the project also involves a thorough evaluation of various ASR models. This evaluation provides a comprehensive overview of the current state of the art in transcription, further contributing to the improvement of automatic subtitling.

1.3.1 Evaluation of state-of-the-art ASR models

To achieve the main goal, one of the steps we have to take is to understand the state of the art in automatic subtitling.

[2] `https://uned.es`

To do this, we evaluate some of the most popular and cutting-edge ASR models that are availible open-souce today, which are listed in Automatic Speech Recognition: State Of the Art. The evaluation takes into account various combination of pre-processing steps (Preprocessing), different datasets (Datasets), and enough metrics (Metrics) to have a precise vision of which models are best suited for speech-to-text tasks. Since ASR models are not inherently designed to generate subtitles, part of the aim of this project is to enhance them with specific capabilities to comply with regulations focused on subtitle quality. Therefore, we also evaluate metrics that measure the extent to which these recommendations are applied to the generated subtitles. All of this is explained in Evaluation Methodology

1.3.2 Python package that generates compliant subtitles

To enhance the quality of automatic subtitling, and to evaluate the ASR models, we need a tool that generates subtitles based on the transcriptions. Taking advantage of this need, a package capable of generate subtitles of high quality which need the minimum human finetuning is developed.

1.4 Contributions to the field

What novel contribution does this master's thesis aim to provide to the field of ASR and subtitling?

It proposes an exceptionally straightforward method for generating high-quality subtitles by leveraging the latest models in ASR. This approach involves a combination of preprocessing and postprocessing techniques that enhance the output and adapt the subtitles to conform to policies that ensure outstanding quality, all of which can be accomplished using a single command, performing as a black box to the final user.

This package should be able to use ASR models to generate subtitles without and with original subtitles as reference, and should be able to evaluate the quality of the generated subtitles. 4

In the array of studies and works examined in the following chapter, maybe Speak2Subs isn't the pioneer in utilizing ASR technology for crafting superior subtitles. However, it's important to note that Speak2Subs either matches or surpasses other models in versatility and broader applicability. In comparison, other models often lack Speak2Subs' breadth, customization options, and adaptability. Unique features of Speak2Subs include Voice Activity Detection, segmentation grouping, a selection from five ASR models, and post-processing aligned with UNE 153010:2012 standards, distinguishing it from other reviewed projects.

Chapter 2

Automatic Speech Recognition: State Of the Art

Automatic Speech Recognition (ASR) and subtitle generation represent important domains within the field of Natural Language Processing, addressing the complex task of converting spoken language into written text. As of the latest developments, the state of the art in ASR and subtitle generation has witnessed remarkable advancements driven by deep learning methodologies, particularly Recurrent Neural Networks (RNNs [6]), Convolutional Neural Networks (CNNs [6]), and transformer-based architectures [19].

ASR systems have evolved significantly, demonstrating top performance owing to the adoption of end-to-end deep learning approaches. Traditionally, ASR pipelines comprised multiple stages involving feature extraction, acoustic modeling, and language modeling. Contemporary systems, however, leverage neural network-based end-to-end models that directly map acoustic signals to transcriptions [8]. This paradigm shift has been facilitated by the success of deep neural networks, such as long short-term memory (LSTM [5]) networks and transformer models, in capturing intricate patterns within sequential data.

Transformer architectures, originally designed for natural language processing tasks, have particularly shown efficacy in ASR due to their ability to model long-range dependencies and contextual information. Models like the Transformer Transducer (Transducer) and the Listen, Attend, and Spell (LAS [3]) architecture have exhibited competitive performance in ASR benchmarks.

Additionally, transfer learning strategies, such as pre-training on vast amounts of unlabeled data followed by fine-tuning on specific ASR tasks, have become prevalent, enhancing the generalization capabilities of ASR models across diverse domains and languages.

2.1 Innovations in Subtitling Technology: An Examination of Recent ASR Research

There are numerous articles and publications related to ASR models. It is true, however, that there are significantly fewer focused on the use of ASR for subtitling purposes. This section provides a concise overview of some particularly intriguing publications that endeavor to progress in the challenging task of automatic subtitling. An overview of these articles and publications can offer an insight into the current state-of-the-art in this field, providing context and reference to objectively assess the advancements of Speak2Subs in the task of automatic subtitling.

It's necessary to differentiate a subtitle from a transcription. A transcription is simply the result of converting audio into text. But a transcription is not a subtitle. A subtitle is adapted to the medium, adjusted for correct reading, with appropriate breaks, size, and the right number of characters. In summary, when we seek to subtitle, transcription is just the beginning. But it's a very important beginning, which is why we need to understand the state of the art of this discipline.

2.1.1 State-of-the-art and recent ASR researchs

Speak2Subs is not the only endeavor to achieve high-quality subtitles. There are various articles that attempt to address this issue. An example is 'Learning to Jointly Transcribe and Subtitle for End-To-End Spontaneous Speech Recognition.' [13] This article presents a very interesting approach to TV subtitling.

This article presents a dual-decoder Transformer model for ASR and subtitling. The model combines an ASR decoder, for verbatim transcription, with a subtitle decoder, both sharing a common encoder. This approach allows the model to effectively utilize TV subtitle data, which are abundant but not verbatim, to improve ASR transcriptions. The model shows improved performance in regular ASR and spontaneous conversational ASR by incorporating the subtitle decoder. It eliminates the need for preprocessing subtitle data, offering a novel way to leverage such data in enhancing ASR systems.

There are additional articles that endeavor to tackle the challenge of subtitling, such as 'Video Subtitle Generation' by Aishwarya Bakale [2] and other co-authors.

The article discusses a system for automatic video subtitle generation in four stages: extracting audio from video, dividing the audio into chunks, recognizing speech using Google's Speech Recognition API, and generating .srt subtitle files. This system is particularly advantageous for people with auditory problems or language barriers. It improves accessibility and understanding of video content, eliminating the need for manual subtitle downloads. The system's effectiveness is highlighted by its accuracy in subtitle generation for various videos.

Another paper, "Automatic Subtitle Generation for Videos" [15], presents a system designed to automatically generate subtitles for video content. The system focuses on improving accessibility for various user groups, including those with auditory challenges or language barriers. It operates through a multi-stage process involving audio extraction from video, speech recognition, and subtitle file creation. The system uses advanced speech recognition tools like CMU Sphinx and DeepSpeech to transcribe audio into text, and then aligns this text with the video to generate subtitles in a .srt format. The study compares the effectiveness of these speech recognition tools and evaluates their performance based on factors like Word Error Rate and system resource usage.

The paper titled "An automatic caption alignment mechanism for off-the-shelf speech recognition technologies" [4] presents a novel system for aligning video captions using standard automatic speech recognition (ASR) applications. The proposed mechanism does not require human-generated transcriptions or specialized software. It involves inserting a unique audio markup into the audio stream before feeding it to an ASR application, and then converting the ASR-generated plain transcript into a timecoded transcript for synchronized caption display. The system is cost-effective and simplifies the captioning process, making video content more accessible, especially for individuals with disabilities or those facing language barriers.

2.1.2 ASR and subtitle compression

As has been previously discussed, transcription and subtitling are distinct processes. A transcription without post-processing is unlikely to serve effectively as a subtitle, due to the necessity of adjusting subtitle length and speed for proper readability. In this work, the criteria set by UNE 153010:2012 are utilized, but there are additional solutions to enhance the accurate reading of subtitles. An example could be the compression of subtitles without loss of information.

The paper titled "Adapting End-to-End Speech Recognition for Readable Subtitles" [11] focuses on enhancing automatic speech recognition (ASR) for subtitling by integrating output compression. It addresses the issue of verbatim transcription reducing readability in subtitles due to screen space and reading time constraints. The study explores both cascaded and end-to-end ASR models, with an emphasis on a Transformer-based end-to-end model fine-tuned with limited data for effective transcription and compression. This model improves accuracy and paraphrasing, producing more concise outputs, and explicitly models length constraints to enhance performance, proving useful for subtitling.

While it is true that this technique is highly beneficial for enhancing the quality and readability of subtitles, it has the clear disadvantage of the subtitles not being a verbatim transcription of what the speaker is saying. Perhaps for academic-themed media, this is an excellent solution, but there are other types of media, such as series and movies, where a non-

literal transcription could lead to a loss of nuances. Therefore, it might be more appropriate to explore other types of post-processing.

2.2 Automatic Speech Recognition models

In the realm of subtitle generation, the advancements in ASR have naturally contributed to improved accuracy and efficiency. Subtitle generation systems typically rely on the integration of ASR with natural language processing techniques for context understanding and grammatical coherence. Transformer-based models, known for their success in sequence-to-sequence tasks, have been adapted for subtitle generation, offering superior contextualization and fluency in transcribing spoken content [12].

It is worth to describe in this section the five Automatic Speech Recognition models that we are evaluating in this thesis, so the lector can have a specific context of what are we evaluating or using in the rest of this document.

2.2.1 Whisper

Textually from their website [1] [14], Whisper is defined as an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web. The use of such a large and diverse dataset leads to improved robustness to accents, background noise and technical language. Moreover, it enables transcription in multiple languages, as well as translation from those languages into English.

Whisper is open-sourcing models and inference code to serve as a foundation for building useful applications and for further research on robust speech processing. It currently have seven different models (tiny, base, small, medium, large, large-v2 and large-v3) depending on the size of the model (i.e., the number of layers, width, heads and parameters).[2]

2.2.2 WhisperX

WhisperX [1] is a model based on whisper, which have important and interesting capacities and improvements out of the box.

WhisperX paper claims that whisper have some problems: The predicted timestamps corresponding to each utterance are prone to inaccuracies, which is bad to generate subtitles. WhisperX is a time-accurate speech recognition system with word-level timestamps that uses two improvements: voice activity detection and forced phoneme alignment. We will see that voice activity detection is implemented in Speak2Subs too, so we can use this feature with any model.

[1] `https://openai.com/research/whisper`
[2] `https://github.com/openai/whisper`

WhisperX proposes a Cut and Merges strategy that improves transcription quality and enables a transcription speedup via batched inference. [3]

2.2.3 NVIDIA NeMo

NeMo from NVIDIA [4] [5] is not limited to ASR. In fact, NeMo is a powerful toolkit for building new state-of-the-art conversational AI models. NeMo has separate collections for ASR, NLP, and Text-to-Speech (TTS) models [9]. Each collection consists of prebuilt modules that include everything needed to train on your data. Every module can easily be customized, extended, and composed to create new conversational AI model architectures.

Focusing in ASR [6], NeMo have a variety of models. Each model has been pre-evaluated, and the best one is the chosen to compare with the rest of ASR.

2.2.4 META Seamless

META Seamless [7] [8] auto-defines as a foundational all-in-one Massively Multilingual and Multimodal Machine Translation model delivering high-quality translation for speech and text in nearly 100 languages. [16]

2.2.5 Vosk

Vosk [9] is very different for the previous ASR models. This speech recognition toolkit does not use superheavy LLM models. In fact, the light models of Vosk are only 50mb each. It is obvious that the precision of the transcriptions can't be close to the rest of the ASR models, but this toolkit have other qualities, like space and velocity. It is added to the comparation to show the enormous difference in terms of subtitle quality are the new wave of ASR models that are emerging right now.

2.3 Conclusion about the state of the art of the subtitle generation task

The listed models belong to highly influential companies within the sector, instilling confidence that these models can perform exceptionally well in transcribing speech to text.

[3] https://github.com/m-bain/whisperX
[4] https://www.nvidia.com/en-us/ai-data-science/generative-ai/nemo-framework/
[5] https://github.com/NVIDIA/NeMo
[6] https://nvidia.github.io/NeMo/asr/models.html
[7] https://ai.meta.com/research/topics/seamless-communication
[8] https://github.com/facebookresearch/seamless_communication
[9] https://alphacephei.com/vosk/

However, as you may have noticed, none are specifically focused on subtitles. Hence, the mere utilization of these models isn't adequate to achieve high-quality subtitles.

Using off-the-shelf ASR models without adjusting their results clearly leads to transcriptions that are not fit for high-quality subtitling. This issue arises partly because these models do not incorporate specific changes in their output to conform it to the standards required for subtitles. Key among these potential alterations are those recommended by the UNE 153010:2012 standards.

Chapter 3

Evaluation Methodology

This master's thesis aims to evaluate Automatic Speech Recognition (ASR) models, focusing on the assessment of their performance and capabilities. The evaluation process involves understanding the significance and challenges associated with the use of ASR models, utilizing various datasets and hyperparameters to facilitate comprehensive analyses, to derive conclusions regarding the strengths and weaknesses of each model, drawing upon a range of metrics obtained through systematic evaluation procedures. Understanding the accuracy, robustness, and overall performance of these models stands as a fundamental objective.

In this chapter, we detail the metrics, datasets, and hyperparameters utilized for evaluation purposes. A series of assessments are conducted, leading to specific conclusions about the models employed in this study.

3.1 Metrics

3.1.1 Error metrics

In many tasks, success is evaluated. However, in this discipline, error is evaluated.

In ASR, error metrics [10] are used to quantify and evaluate the accuracy and performance of the system in transcribing spoken language into text. Several key error metrics and variants are commonly used in ASR evaluation.

There are some concepts that need to be explained before getting into specific metrics:

A **substitution** occurs when the ASR system incorrectly recognizes a word, replacing it with a different word. For example, if the correct word in the speech was "cat" but the ASR system transcribed it as "cap," this would be counted as a substitution error. Substitutions can occur due to various factors like the speaker's accent, background noise, or the system's limitations in understanding the context.

A **deletion** error happens when the ASR system fails to recognize and transcribe a word that was spoken. For instance, if the original speech was "the quick brown fox" and the ASR system transcribed it as "the brown fox," the word "quick" has been omitted, resulting in a

deletion error. Deletions can be caused by speech that is too quiet, fast, or unclear for the system to detect accurately.

An **insertion** error occurs when the ASR system adds a word that was not present in the original speech. For example, if the actual spoken phrase was "go home," but the system transcribed it as "go right home," the word "right" is an extra word that was not spoken, marking it as an insertion error. Insertions can happen due to background noises being misinterpreted as speech or the system misinterpreting speech patterns.

The metrics described in the following paragraphs are commonly used for several reasons. They are simple, intuitive, and easy to understand. They are consistent in comparison, meaning they allow for direct comparison between models without the need for adjustment. They are easy to implement. And, of course, they are the standard metrics that have been used in this field for the last few years.

WER - Word Error Rate

WER [10] measures the ratio of the total number of errors (substitutions, deletions, insertions) in the recognized output compared to the reference (ground truth) transcript.

$$WER = \frac{(S + D + I)}{N}$$

S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of words in the reference transcript. Lower WER indicates higher accuracy.

MER - Match Error Rate

Evaluates the alignment between recognized speech output and the reference transcription at the word level. MER [10] assesses the quality of the alignment by measuring the discrepancy between the recognized transcript and the ground truth reference transcript.

$$MER = \frac{(S + D + I)}{(N + M)}$$

Where S is the number of substitutions, D is the number of deletions, I is the number of insertions, N is the total number of words in the reference, and M is the total number of words in the recognized transcript.

WIL - Word Information Lost

Measures the loss of information during the transcription process, specifically focusing on the content of the recognized words compared to the reference or ground truth transcription.

$$WIL = \frac{(D + S)}{(N + I)}$$

where D is the number of deleted words, S is the number of substituted words, N is the total number of words in the reference, and I is the number of inserted words in the recognized transcript.

WIL quantifies the loss of information due to deletion and substitution errors in the recognized transcript compared to the reference transcript. Higher WIL values indicate a higher degree of information loss, meaning that a larger proportion of words in the recognized output are either substituted or missing compared to the reference. WIL complements metrics like Word Error Rate (WER) and Character Error Rate (CER) by focusing specifically on the informational content lost during transcription. It can provide insights into the overall fidelity of the transcribed output in terms of preserving the original information from the spoken utterance.

WIP - Word Information Preserved

Measures the proportion of correctly transcribed words in the recognized output compared to the reference or ground truth transcription, focusing on the retention or preservation of accurate information..

$$WIP = \frac{N - (D + S)}{(N)}$$

where N is the total number of words in the reference, D is the number of deleted words, and S is the number of substituted words in the recognized transcript.

WIP quantifies the proportion of correctly transcribed words in the recognized transcript compared to the reference transcript. A higher WIP value indicates a higher preservation of accurate information, meaning that a larger proportion of words in the recognized output match the reference.

WIP is a useful metric in evaluating ASR systems as it provides insights into how well the system preserves the original information conveyed in the spoken utterance by accurately transcribing the words. It complements metrics like Word Error Rate (WER) and Character Error Rate (CER) by focusing specifically on the preservation of information rather than overall error rates.

3.1.2 Weighted error metrics

WWER - Weighted Word Error Rate

Using a standard formula like WER may overlook how different errors affect the overall outcome. Some errors might be more disruptive or harder to rectify than others. Also, the formula doesn't distinguish between specific types of errors; for instance, it can't tell if an error comes from a substitution or a combined deletion and insertion.

Hunt, in 1990 and which paper seems is not in public repositories [7], proposed a solution by introducing a weighted accuracy measure. In this approach, errors resulting from substitutions are considered the most crucial and are given full weight (a value of 1). However, errors due to deletions and insertions are seen as less impactful and are weighted at 0.5 each. This means that while substitution errors are fully accounted for in the evaluation, deletions and insertions are considered less severe, receiving only half the weight of a substitution error. This weighted approach helps prioritize the importance of different error types in assessing the accuracy of speech recognition systems.

$$WER = \frac{(S + 0.5D + 0.5I)}{N}$$

WMER - Weighted Match Error Rate

There is no specific paper defining WMER for the first time, but seems logic to use the same principle as with Weighted WER, as long as it calibrates different error types.

$$WMER = \frac{(S + 0.5D + 0.5I)}{(N + M)}$$

3.1.3 Normalized error metrics

In the context of measuring word error rate (WER) or other evaluation metrics in natural language processing tasks, the use of uppercase and lowercase letters, punctuation marks, as well as the presence of periods and commas can negatively impact precision. This is why normalization is often necessary before conducting such evaluations.

By doing so, the evaluation process becomes more robust and less sensitive to superficial differences in capitalization, punctuation, and other non-semantic aspects of the text. This allows for a more reliable assessment of the model's performance based on its ability to produce correct and meaningful content.

The normalized metrics are named like the original metrics, with the 'N' prefix: NWER, NMER, NWIP, NWIL, NWWER and NWMER.

3.1.4 Mismatch metric

Mismatch metrics are specific to Speak2Subs and are designed to estimate the accuracy of models in predicting correct timestamps for each word. When the beginning of an estimated word falls within the timestamp range of a subtitle, but the end of the word falls within the timestamp range of the next subtitle, it signifies that the model lacked precision in estimating the correct timestamps. The mismatch ratio indicates the percentage of words assigned to the subtitle before or after the correct one. Naturally, for this metric to be meaningful, the timestamps of the original subtitle must be precisely defined with high accuracy.

There is no specific formula to this metric, instead the algorithm is explained in Error metrics.

3.1.5 NER metrics

The NER metric specifically evaluates the performance of an ASR system in accurately recognizing and transcribing named entities from spoken language. This metric assesses how well the ASR system can capture and correctly transcribe important entities mentioned in the audio content. Specifically, NER measures the impact of edition mistakes (same word but with differences) and recognition mistakes (different word)

$$NER = \frac{(N + E + R)}{N}$$

Where N is the total of words, E is the total of editions, and R is the total of recognitions. The algorithm to calculate the NER metric is explained in chapter 4.

3.1.6 Execution time metrics

Precision measures are not the only measures that should be taken into account. Measuring execution time is crucial in various fields of computing and software development for several reasons. If we want subtitles in realtime in a streaming, we can't wait too long to get the subtitles.

Execution time is measured in seconds. VAD and segmentation are not being considered because it does not take a significant amount of time.

3.1.7 Compliance metrics

Generating subtitles involves adherence to compliance standards to ensure quality, accessibility, and interoperability. Compliance standards serve as a framework to regulate the creation of subtitles, addressing issues such as accuracy, synchronization, and readability.

This project selectively incorporates applicable guidelines from the UNE (Spanish Association for Standardization) to establish a framework for its implementation. The UNE 153010:2012 [18] has a wide set of guidelines which, if followed, ensure that the subtitles have the correct format and quality.

There are three guidelines of this framework that Speak2Subs is able to check in the VTT files: Number of characters per line, number of lines, and number of characters per second.

UNE 153010:2012 4.3 - Number of Lines of Text

The subtitles should occupy a maximum of two lines of text.

UNE 153010:2012 4.6 - Number of characters per line

The maximum character limit per line should be 37.

UNE 153010:2012 5.1 - Subtitle text exposure speed

The exposition speed of the subtitle text should follow the rhythm of the original and facilitate comfortable reading. Normally, the maximum number of characters per second would have to be 15 cps (characters per second).

3.2 Preprocessing parameters

Evaluating Automatic Speech Recognition models involves not only comparing different models but also understanding the influence of tuning various parameters within the tools or packages utilizing these models. This is the case of Speak2Subs, which have some preprocessing parameters that affects directly the model results. Therefore, evaluating the impact of different preprocessing parameter configurations within Speak2Subs is crucial to understanding the sensitivity and robustness of the models across various settings.

Moreover, tuning preprocessing parameters enables the optimization of ASR models for specific tasks or datasets. Different datasets may require distinct preprocessing parameter configurations due to media duration, importante of the speech context, variations in language, accents, background noise, or recording conditions.

Documenting the impact of preprocessing parameters contributes to establishing best practices and guidelines. These findings serve as valuable references for practitioners and researchers, guiding them in developing or fine-tuning ASR models effectively.

In conclusion, while comparing different ASR models remains fundamental, examining the impact of diverse preprocessing parameter settings within Speak2Subs is equally crucial.

Therefore, tests have been conducted with different combinations of preprocessing parameters, which are listed below. A more deep explaination of every preprocessing parameter can be found in chapter 4.3.2.

3.2.1 Voice Activity Detection

Voice Activity Detection (VAD) [1] refers to the process of identifying segments within an audio signal where there is significant speech activity or the presence of a human voice. The primary goal of VAD is to distinguish between speech and non-speech portions in an audio, enabling more efficient processing and analysis by ASR systems.

[1] `https://en.wikipedia.org/w/index.php?title=Voice_activity_detection&oldid=1177185607`

3.2.2 Segmentation

As previously mentioned, Voice Activity Detection (VAD) extracts segments from the audio where speech is detected, with a typical segment corresponding to a phrase. If we want to not use VAD and still get segments, we can use this preprocessing parameter.

With segmentation enabled, manual segmentation is performed, using the “max speech duration” preprocessing parameter as a reference, generating a segment every “max speech duration” seconds.

On the contrary, if neither VAD nor manual segmentation is desired, the result is a single segment encompassing the entire audio duration.

3.2.3 Group Segments

By default, segments are organized in groups (see 4.3.2), and the ASR models uses each segment group as a batch. If we don’t want to group the segments, and use each individual segment, which uses to be a single sentence, as a batch for the ASR models, we can enable this preprocessing parameter.

3.2.4 Max speech duration

Each group of segments must have a specific duration. The Max Speech Duration determines the maximum duration (in seconds) of each segment group, aiming to create these groups with a duration as close as possible to this preprocessing parameter without exceeding it.

3.2.5 Noise reduction

In the context of audio, noise reduction refers to the systematic process of minimizing or eliminating unwanted sounds, often characterized as “noise" from an audio recording

3.3 Datasets

To evaluate the various ASR models, five datasets have been employed, each containing a series of videos accompanied by their original subtitles. These datasets originate from different classes or informative materials at UNED, but it should be noted that they are not necessarily similar. The datasets incorporate diverse elements such as male and female voices, distinct accents, and speech patterns.

One common factor among the five datasets is that they all comprise content in the Spanish language. Theoretically, the language used should not significantly impact the models, assuming they are balanced in this aspect. Nevertheless, this aligns favorably with one of the motivations behind this project—to utilize this software for the automatic translation of UNED materials.

The five datasets are the following:

3.3.1 Materiales digitales accesibles

1. Code: mda
2. Eight media files
3. Same male voice for all eight media files.
4. Total aproximate length of 55 minutes

The voice belongs to a male with an Andalusian accent. This dataset serves as an intriguing test to assess if ASR models can comprehend Spanish in all its variants and dialects. This dataset presents a particular challenge as the Andalusian dialect isn't as straightforward to understand as standard Castilian Spanish. [2]

3.3.2 Accesibilidad en la atención a clientes

1. Code: atc
2. Three media files
3. Each media file has a different female voice.
4. Total aproximate length of 22 minutes

It is worth noting that the female voice in the asset "atc_3.mp4" exhibits some speech difficulties, posing a challenge for ASR models. Overcoming this challenge is of interest to ensure accessibility and captioning for voices of any kind. [3]

3.3.3 Cómo formar sobre diseño para todas las personas

1. Code: cdp
2. Four media files
3. Same female voice for all four media files.
4. Total aproximate length of 23 minutes

The dataset contains videos featuring a female voice without any distinctive attributes. This dataset could be viewed as relatively straightforward for ASR models to interpret and comprehend. [4]

[2] https://canal.uned.es/series/5a6fa618b1111fd8068b4569
[3] https://canal.uned.es/series/5d9335f1a3eeb019478b456a
[4] https://canal.uned.es/series/61964058b6092324812d9235

3.3.4 MOOC Discapacidad y Defensa Legal Activa en la Era Digital

1. Code: ddl
2. Five media files
3. Different voice for each media file.
4. Total aproximate length of 50 minutes

Each media file of the dataset features a different individual. These recordings encompass a variety of tones and nuances, including both male and female voices. [5]

3.3.5 Móviles accesibles para todos

1. Code: mat
2. Four media files
3. Different voice for each media file.
4. Total aproximate length of 36 minutes

Two female voices and one male voice are included in the dataset. One of the female voices exhibits some speaking difficulties, challenging the models in terms of accessibility for all users. [6]

3.4 Configurations

As mentioned in previous sections, five ASR models are evaluated: OpenAI Whisper, WhisperX, META Seamless, NVIDIA NeMo, and Vosk. To measure the potential of these models, they undergo testing against a battery of datasets with different hyperparameter combinations. The different parameter combinations are referred to as configurations.

Certain preprocessing parameters remain consistent across all configurations. These include Noise Reduction, by default, disabled, and Max Speech Duration, by default, set to 30 seconds.

In total, there are four configurations to be applied.

1. Conf 1. Default
2. Conf 2. No VAD
3. Conf 3. Sentences
4. Conf 4. Raw audio

[5] `https://canal.uned.es/series/5ddfabea5578f2263425df27`
[6] `https://canal.uned.es/series/5a6f381db1111fac3a8b4569`

3.4.1 'Default' configuration

- VAD: ON
- Segments: ON
- Group segments: ON

Configuration 1, 'default,' employs default preprocessing parameters. It incorporates Voice Activity Detection, splits the audio file into small speech segments, and groups them into segment clusters. Since the maximum speech duration is kept at its default, these clusters will have a total duration close to 30 seconds.

The goal of this configuration is to measure the models with optimal preprocessing: VAD enhances speed by eliminating silences, improving precision by preventing the models from confusing other sounds with speech. The grouping of segments, by creating longer batches with more information, assists the models in having sufficient context in each phrase, provided they can leverage it. Therefore, the results with this configuration should be optimal.

3.4.2 'No VAD' configuration

- VAD: OFF
- Segments: ON
- Group segments: ON

Configuration 2, or the 'No VAD' configuration, aims to assess the impact of applying Voice Activity Detection (VAD) to ASR models. As the remaining preprocessing parameters are kept at their default settings, we can gain a precise understanding of the actual impact of VAD, both in terms of speed and quality. Instead of grouping speech segments, each segment group is a 'Max Speech Duration' segment, which is 30 seconds by default.

3.4.3 'Sentences' configuration

- VAD: ON
- Segments: ON
- Group segments: OFF

Configuration 3, or 'sentences' configuration, is akin to the default setting, except for the notable distinction that it does not group segments. Therefore, given the application of Voice

Activity Detection (VAD), models will need to perform speech-to-text on a sentence-by-sentence basis, without grouping them and lacking extensive context. The primary objective of this configuration is to assess the importance of context in transcription. If the results do not deteriorate compared to the default configuration, it suggests that the models suffice with listening to the individual sentence for transcription and do not require information from preceding or subsequent sentences. Conversely, if there is a loss of precision, it implies that models benefit from context, and consequently, longer batches improve the results.

3.4.4 'Raw audio' configuration

- VAD: OFF
- Segments: OFF
- Group segments: OFF

Configuration 4, or the 'raw audio' configuration, aims to assess models using untreated audio. Without applying Voice Activity Detection (VAD), segmentation, or any other pre-processing, this configuration solely evaluates the models' capability to handle unprocessed, long-duration audio. Some models may struggle with audio of such extended length, and this, too, is a factor under evaluation.

3.5 Models

For each ASR tool, multiple models are available. In this evaluation, an effort has been made to select the best models from each company. The chosen models are as follows:

- **Whisper**: openai/whisper-large-v3
- **WhisperX**: large-v2
- **Seamless**: facebook/seamless-m4t-v2-large
- **Vosk**: vosk-model-es-0.42
- **NeMo**: stt_es_conformer_ctc_large

These models have been chosen to represent each ASR tool in the evaluation, aiming to utilize the most effective options available from each respective company. NeMo is a special case, because it have a wide list of models. The chosen one is the model that best performed in tests.

3.6 Metodology conclusions

Given the various combinations of hyperparameters, datasets, and models, it can be asserted that the evaluation methodology is sufficiently rigorous and comprehensive to yield satisfactory results, allowing little room for error.

Chapter 4

Speak2Subs package

In this chapter everything about the package developed in this master's thesis is going to be fully explained. How to install it, what requirements are necessary, how to use it, what hyperparameters are tuneable, etc.

Of course, there is a huge section explaining how does it work, what is happening inside and what algorithms uses. Usually, code has been avoided, but some pseudo-code is necessary to make some explanations clear.

In previous chapters it has been briefly described that this package has several features. This chapter attempts to describe each of them so that the objective, use and internal functioning of each of the features is understood.

A resumed view of the use cycle can be seen in Figure 4.1.

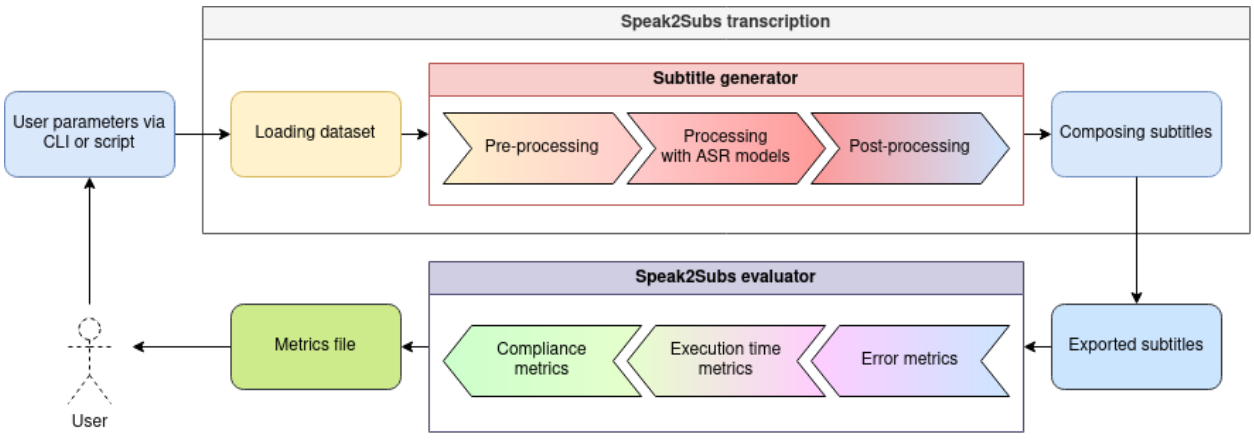

Figure 4.1: High level diagram of the use cycle

4.1 Features

Speak2Subs has three fundamental features: subtitle generation based on a reference template, subtitle quality evaluation, and subtitle generation without a reference template. This

section provides a concise introduction to each of these features, while subsequent sections describe the installation process, usage, and internal mechanics in more detail.

4.1.1 Subtitle generation based on a reference template

One of the features is to generate subtitles. To correctly evaluate the ASR models on the table, we need to put them to the test and generate captions from datasets. But it is not enough to have subtitles, for the evaluation to be viable we need to compare them with the already original, presumably correct, subtitles. This implies that the generated subtitles must have the same timestamps as the original subtitles.

To make it clear, if our original subtitle contains the following entry:

```
[00:00:02.000] -> [00:00:06.000] Buenos días, esto es una prueba.
```

Our generated subtitle could not have the following structure:

```
[00:00:02.000] -> [00:00:03.000] Buenos días
[00:00:04.000] -> [00:00:06.000] Eso es una rueda.
```

Because when it comes to comparing texts, there is clearly no match. Obviously by restructuring timestamps we obtain the equivalences:

```
[00:00:02.000] -> [00:00:06.000] Buenos días, esto es una prueba.
[00:00:02.000] -> [00:00:06.000] Buenos días, eso es una rueda.
```

In conclusion, to evaluate subtitles we need to previously load the original subtitles as a template. So, one of the usage of this package, is to **generate subtitles using a template.**

As a disadvantage of using templates, the package is limited to the original structure, and may not be able to adjust the compliance parameters as it would in templateless generation.

4.1.2 Subtitle quality evaluation

This package allows you to evaluate the quality of a subtitle file based on a reference file. Although this feature is what has been used for the evaluation phase, it is not necessary that the subtitles to be evaluated have been generated by this same package. The only requirement is that both files share timestamps and number of entries.

4.1.3 Subtitle generation without a reference template

Of course, you can also generate subtitles without a base template. In fact, it is the functionality that gives value to the package beyond just as an evaluator.

With this functionality, the grouping of the transcribed words (tokens) into sentences and, in turn, subtitles, must follow the compliance policies described in the evaluation section. For this, an algorithm is used that is explained later.

A contraindication of these subtitles is that we cannot evaluate the error metrics, but we do have information on generation time metrics and compliance with the UNE.

4.2 Installation

The package Speak2Subs is programmed in Python 3.10. It is available on GitHub [1].

Before installing the package, some requisites are needed.

4.2.1 Install PyTorch

PyTorch is an open-source machine learning library used for various tasks like natural language processing, computer vision, and more. It provides tools for building and training neural networks, offering flexibility and efficiency in experimentation due to its dynamic computational graph feature. Speak2Subs needs Torch to be installed.

The command must be adjusted to your requirements [2]

4.2.2 Install Docker

Docker is a platform designed to make it easier to create, deploy, and run applications using containers. Containers allow developers to package up an application with all the necessary parts (such as libraries and other dependencies) and ship it as a single unit. Docker provides a way to automate the deployment of applications inside containers, ensuring consistency across different environments, from development to testing and production. Since Speak2Subs uses containerized ASR models, Docker needs to be installed [3].

```
1 $ pip install docker
```

4.2.3 Install Speak2Subs

Finally, Speak2Subs can be installed [4]. The package has a few more dependencies, but they are automatically installed at the same time.

[1] https://github.com/JulioFresneda/Speak2Subs
[2] https://pytorch.org/get-started/locally/
[3] https://docs.docker.com/engine/install/
[4] https://github.com/JulioFresneda/Speak2Subs

From Pypi

```
$ pip install speak2subs
```

From source code

Alternatively, this command pulls and installs the latest commit from this repository, along with its Python dependencies:

```
$ git clone https://github.com/JulioFresneda/Speak2Subs.git
$ cd Speak2Subs
$ pip install -e .
```

4.3 Usage

This package can be used directly via CLI, and in a Python script. You can look for usage examples in the GitHub page.

This package can be utilized both via the command line and within a Python script, for both generating subtitles and evaluating them. The package design is oriented towards a dataset style, meaning it defaults to working with folders of files. However, it can also be specified to generate or evaluate a specific file. The following sections will explain specific examples.

4.3.1 How to generate subtitles

Speak2Subs supports MP4 and WAV files. When using MP4, it automatically converts them to WAV format. Optionally, if original VTT files are available, similar timestamped subtitles can be generated based on the original VTT as a template. To generate subtitles, all MP4 or WAV files (and VTT files if desired) should be in the same folder. If VTT files are present, they must share the same name as the media file, disregarding the extension. If no dataset name is specified, the name of the folder containing the files is used. To export the results, a folder needs to be specified where the generated VTT files will be exported.

Individual files are compatible too, the only difference in usage is that instead of using a path for the dataset folder, a path for the media file is needed

Command Line Interface usage

If you want to generate subtitles

```
  $ speak2subs --media_path="./mydataset" --export_path="./results"
```

You can choose the ASR models to use. Default is whisperx.

```
$ speak2subs -mp="./mydataset" -ep="./results" --asr="nemo, whisper"
```

If you want to generate subtitles and use original VTT as reference

```
$ speak2subs -mp="./mydataset" -ep="./results" --use_vtt_templates
```

If you want to generate subtitles for a particular file

```
$ speak2subs -mp="./mydataset/media_1.wav" -ep="./results" --use_vtt_templates
```

If you want to get the full list of arguments

```
$ speak2subs --help
```

Python usage

It is as easy as with CLI to use this package in a python script.

```python
from Speak2Subs import speak2subs

speak2subs.transcript('./mydataset',
                      export_path='./results',
                      asr='all',
                      use_vad=True,
                      segment=True,
                      group_segments=False,
                      max_speech_duration=30,
                      use_vtt_template=True,
                      reduce_noise=False)
```

4.3.2 How to evaluate subtitles

If subtitles have been previously generated, using the original subtitles as a reference, it is possible to evaluate the outcomes without effort. Much like the process for generating results, the original media folder is required to read the original subtitles, alongside the results folder to read the generated subtitles. The sole distinction lies in the additional parameter 'evaluate' that must be utilized. As straightforward as that. The result is an Excel file with all the metrics evaluated.

Command Line Interface usage

First we generate subtitles

```
$ speak2subs --media_path="./mydataset" --export_path="./results" --asr="seamless, vosk"
```

Then we evaluate them

```
$ speak2subs -mp="./mydataset" -ep="./results" --evaluate
```

We can evaluate a pair of VTT too

```
$ speak2subs --evaluate --ref_vtt_path="./reference.vtt", --
 pred_vtt_path="./predicted.vtt"
```

If we chose to evaluate a pair of VTT instead of the results of a generation, instead of an Excel with metrics, the result is an output on the terminal.

We can use more arguments, but it won't take effect because the rest of them are focused on the generation task.

Python usage

You can use it in python too.

```python
from Speak2Subs import speak2subs
from Speak2Subs import speak2subs

# If we want to evaluate the generated subtitles for our dataset
speak2subs.evaluateFolder( "./mydataset", "./results")
# If we want to evaluate a specific pair of subtitles
result = speak2subs.evaluatePair("./media_1.vtt", "./media_1_PRED_.vtt")
# If we want to evaluate the compliance
compliance = speak2subs.evaluateCompliance("./media_1.vtt")

```

4.4 How does it work: Subtitle Generator

This section attempts to explain the internal gears of the package. How to install it and how to use it has already been explained, but until now it has not been detailed what is done in each step.

To generate subtitles, Speak2Subs does not limit itself to using the capabilities of ASR models. We applied a very interesting pre-processing, and we left open the possibility of a certain quite promising post-processing that perhaps could be implemented in the future.

In Figure 4.2 we see a global outline of what happens when generating subtitles. The following sections detail what happens in each piece of the puzzle in depth.

4.4.1 Loading Dataset

Before starting the generation process, Speak2Subs needs to know the location of the files to transcribe. Given that complex operations will be performed, simply loading the audio

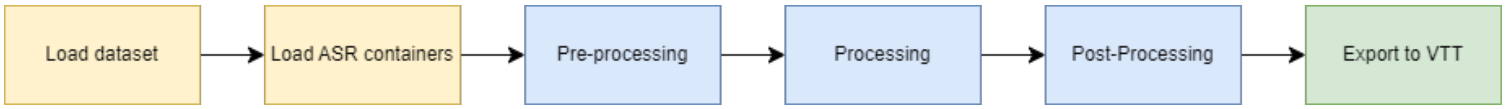

Figure 4.2: Generation pipeline

files is not sufficient.

Speak2Subs utilizes a series of classes to organize these files into objects.

The direct equivalent of a .wav or .mp4 file to be transcribed is the object of type Media. This object contains crucial metadata of the file, such as its path, name, path of the original subtitles, segment groups, and, upon completion of the process, the generated subtitles. Later on, an explanation will be provided for what segments and segment groups are and their respective purposes.

A Dataset is, therefore, a collection of Media objects. As explained in the usage of Speak2Subs, one of the necessary parameters for subtitle generation is the directory of the files. During the loading of datasets, the package detects all .wav, .mp4, and .vtt files.

Once detected, based on the file name, it assigns the original subtitle to each .wav file (if it exists and has been requested to be loaded). If there is no .wav but a .mp4 exists, a pre-transformation to .wav is performed.

In conclusion, during the loading phase, information is obtained for each audio file, with a highly object-oriented approach.

4.4.2 Pre-processing

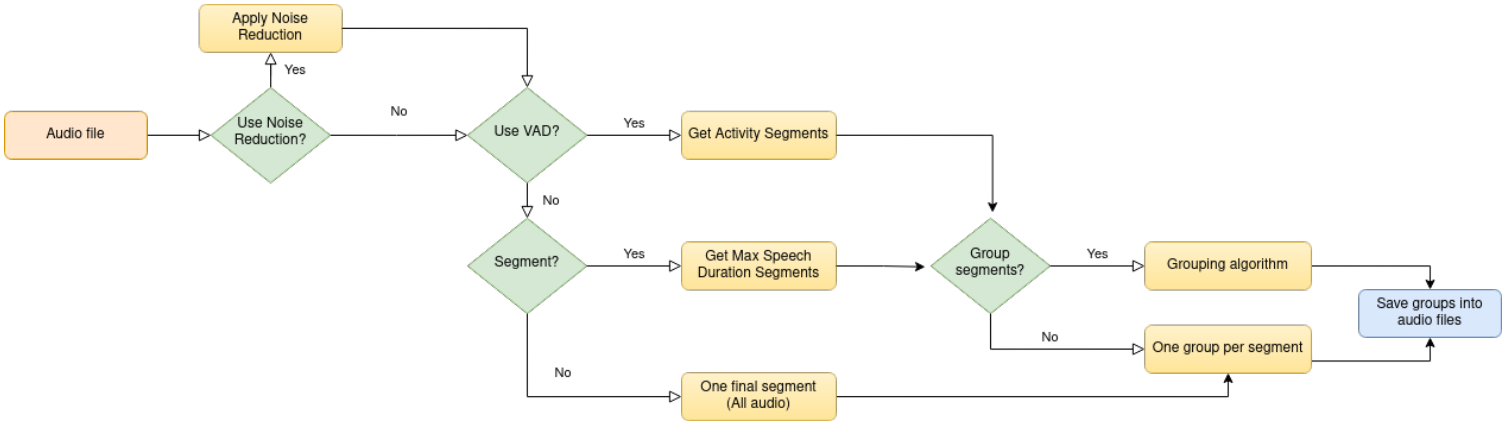

Figure 4.3: Pre-processing pipeline

The data preprocessing serves a purpose beyond preparing data for proper model utilization. It is crucial for ensuring that the input data is in a format that models can effectively utilize, thereby enhancing their performance. Proper preprocessing addresses issues such as noise, irrelevant information, and varying data formats, ultimately improving the model's ability to extract meaningful patterns and insights.

If this project were solely focused on model evaluation, this phase would be much simpler; however, as previously mentioned, model evaluation is not the only objective of this project. We also aim to generate high-quality subtitles. This necessitates the implementation of various preprocessing capabilities designed to get superior results.

These capabilities, briefly outlined in other sections, include noise reduction, Voice Activity Detection (VAD), segmentation, and grouping. Each of these capabilities will now be described in more detail. A resumed step-by-step guide can be seen in Figure 4.3.

Each step is going to be explained in the following sections.

Noise Reduction In the context of audio, noise reduction refers to the systematic process of minimizing or eliminating unwanted sounds, often characterized as "noise" from an audio recording. This unwanted noise can originate from various sources, such as background disturbances, electronic interference, or environmental factors, and it can significantly affect the overall quality of the audio signal.

Noise reduction techniques in audio processing involve the application of algorithms or filters to identify and reduce the amplitude of undesirable sounds while preserving the integrity of the desired audio content. These techniques aim to enhance the signal-to-noise ratio, resulting in clearer and more intelligible audio recordings.

Thankfully, there are plenty of Python libraries and packages that can do this complex task for us. In this case, Speak2Subs uses the noisereduce [5] package.

The impact on error metrics resulting from the activation of this capability has been evaluated, and surprisingly, it has been observed that the impact is negative. This is because, even in the presence of significant background noise, Automatic Speech Recognition models can discern words from noise. In contrast, when noise reduction is applied, as it is not perfect, it also affects the voices, rendering some words unintelligible. That is why it is disabled by default, but is available in case it might be useful in a specific scenario.

Voice Activity Detection

Voice Activity Detection (VAD) refers to the process of identifying segments within an audio signal where there is significant speech activity or the presence of a human voice. The primary goal of VAD is to distinguish between speech and non-speech portions in an audio, enabling more efficient processing and analysis by ASR systems.

VAD plays a crucial role in improving the performance and accuracy of ASR models by reducing the amount of irrelevant information and noise introduced during non-speech intervals. By accurately detecting speech segments, the ASR system can focus its resources on transcribing the meaningful spoken content, leading to better overall recognition results.

[5] `https://github.com/timsainb/noisereduce`

This capability, which, by the way, WhisperX provides out of the box, not only has implications for accuracy but also significantly enhances subtitle generation speed, as demonstrated in our evaluation. That's why it is enabled by default in Speak2Subs.

The VAD backend used is Silero VAD [6]

Segments

A Segment is, quite redundantly, a portion of the file. Its metadata has a start timestamp, an end timestamp, and upon completion of the process, predicted subtitles for this segment.

It is crucial to divide the file into segments because some ASR models lack native capacity to process lengthy audio recordings. That's why it comes activated by default. In fact, WhisperX employs this very mechanism.

As previously mentioned, Voice Activity Detection (VAD) extracts segments from the audio where speech is detected, with a typical segment corresponding to a phrase. But what if we opt not to use VAD yet still need the audio in segments?

In such cases, manual segmentation is performed, using the "max speech duration" hyperparameter as a reference, generating a segment every "max speech duration" seconds.

On the contrary, if neither VAD nor manual segmentation is desired, the result is a single segment encompassing the entire audio duration.

Segment groups

Dividing an audio into segments proves more manageable and efficient for ASR models, but to what extent? If the segments are too small, such as having one per sentence, ASR models may lose crucial information related to the context of the conversation. In fact, this aspect is reflected in the evaluation.

Therefore, while segmenting audio is beneficial, it is also crucial to ensure that these segments have a balanced duration. This is why we employ the segment grouping algorithm.

As illustrated in the pseudo-code, the algorithm aims to form groups of segments with a total duration as close as possible to the MaxSpeechDuration hyperparameter, always without exceeding this specified duration.

Once these segment groups are obtained, an audio file is saved per group in a local cache directory. These groups will be utilized by ASR models for transcription. Naturally, the original timestamps are retained to synchronize the generated subtitles with the original audio.

4.4.3 Processing

Once Speak2Subs generates separate audio files in shorter tracks, formed by groups of segments (or a single track, depending on hyperparameters), the phase of using ASR models

[6] `https://github.com/snakers4/silero-vad`

Algorithm 1 Group segmentation algorithm

```
1: procedure GroupSegments(sentences)                ▷ Generates groups of segments
2:     SegmentGroups ← []
3:     CurrentList ← []
4:     for sentence ∈ sentences do
5:         DurationSoFar ← ∑(sg.end − sg.start), sg ∈ CurrentList
6:         if DurationSoFar + sentence.duration ≤ MSD then        ▷ Sentence fits in
7:             CurrentList[] ← sentence
8:         else                                                   ▷ Group full
9:             SegmentGroups[] ← CurrentList
10:            CurrentList = [sentence]
11:        end if
12:    end for
13:    return SegmentGroups
14: end procedure
```

begins.

ASR Containers

In the first graph of this chapter, it can be observed that even before preprocessing, there is a stage where ASR containers start running. This has a reason. Each ASR model is from a different company, developed under different conditions, and, of course, requires different prerequisites and packages. Some models don't even use similar versions of Python and Torch.

This problem is easily addressed by using containers [7]. For each ASR model, we have a Dockerfile that installs all the prerequisites and libraries, and there's an image hosted on DockerHub (https://hub.docker.com/u/juliofresneda). These images come pre-installed with ASR and the necessary scripts to obtain subtitles. When Speak2Subs needs the services of an ASR model, it downloads the corresponding image and launches a container, which remains active until its task is completed.

These containers have a shared volume with the local host, so there's no need to copy the audio files to the container or receive a response from it. Everything works on this volume, housed in the temporary cache folder generated by the package.

It's important to note that while some models are lightweight, like Vosk, there are models that require a substantial amount of RAM. Seamless, for instance, may require more than 16GB of RAM, and Docker may terminate the process if there isn't enough available.

With the containers ready to receive commands, Speak2Subs starts sending them. For each ASR to be used, and for each Media file (audio), the ASR model performs its speech-to-text operation for each segment group. The result is written to the shared volume.

[7] `https://www.docker.com/resources/what-container/`

Algorithm 2 Processing phase algorithm

```
    procedure GroupSegments(sentences)                    ▷ Processing with ASR
2:      for asr ∈ asrlist do
            for media ∈ dataset do
4:              for segmentgroup ∈ media do
                    ASR(asr, segmentgroup)                ▷ Run in container
6:              end for
                CompileSubtitles(media, results)
8:              ExportSubtitles(media, results)
            end for
10:     end for
    end procedure
```

The pseudocode is not completely precise because the iteration of each group of segments is actually carried out within the container itself, but the objective is to clarify the concept.

When the container finishes with all Media files, it concludes.

Except for Seamless, the result is a list of transcribed words, along with the local timestamps of each segment group. How these results are handled is explained later, but it can be advanced that it is not a trivial task.

Compile Subtitles

The keen-eyed readers may have noticed a detail: If ASR models transcribe each segment group separately, what happens with the timestamps provided by the ASR for each word? The model lacks context of the original audio, so it will return "local" timestamps, where the first word begins at second 0 (local), even if that's not the case in the original audio. Not only that, but what about the silences we deleted with VAD? This scenario is a bit complex, so maybe it will be easier to grasp with figure 4.4.

There are two main problems to solve: keeping track of silences between segments and keeping track of the silences between segment groups. To convert local timestamps into global timestamps applicable to the original audio file, Speak2Subs applies the following algorithm.

For each group of segments, iterate through the list of tokens (predicted words with local timestamps). For each token, iterate through the list of segments of the current group. Keep a count of seconds of silence since the start of the segments iteration. Silences can be calculated by subtracting the end timestamp of the last segment from the start timestamp of the current segment. Now, we can translate from local to global. Add the count of the silence plus the start of the segment group (which is a real timestamp) to the local start and end timestamps.

If these local timestamps fit between the start and end of the segment, it means that the token (word) belongs to this current segment, so we can restart the count and move on to

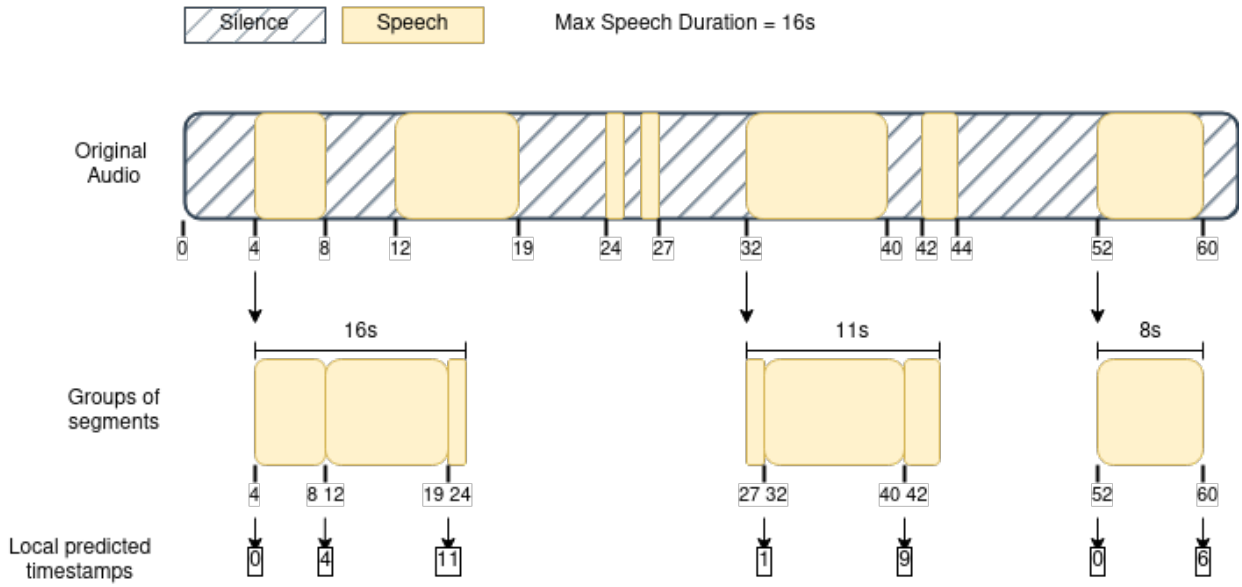

Figure 4.4: Understanding local timestamps

the next token.

The reality is that the algorithm described in the pseudocode differs from the actual implementation in one thing. This discrepancy arises from the fact that the predicted timestamps for each word are not entirely accurate. In some situations, it may appear that a word starts in one segment and ends in the next. To address this issue, the 'if' statement in line 13 only takes 'globalStart' into consideration and not 'globalEnd'. The 'mismatch' metric quantifies the impact of this issue on the predicted subtitles.

Seamless, by Meta, poses a challenge. Its timestamp predictions are not at the word level but rather at the sentence level. To navigate this obstacle, attempts have been made to estimate the timestamps for each word within the sentences, taking into account the number of characters, commas, and periods. While it may not be an extremely precise estimation, there is no better alternative solution at the moment.

Once we have all the words with the actual timestamps, the goal of generating subtitles synchronized with the audio track is much closer. However, there are still a few extra steps to fulfill the mission.

Export to VTT

Once the word predictions with their associated actual timestamps are obtained, the last step remaining for creating subtitles in VTT format is determining how to order these words. For this purpose, Speak2Subs offers two paths.

If, for some reason, we already have VTT files [8] associated with the media we want to subtitle and we want the generated VTT files to share the same time structure, we simply

[8] `https://developer.mozilla.org/en-US/docs/Web/API/WebVTT_API`

Algorithm 3 Local to global timestamps

```
    procedure LocalToGlobalTimestamps(tokenLists, media)
        for segmentGroup ∈ media.segmentGroups do
 3:         groupStarts ← segmentGroup.start
            tokenList ← tokenLists[segmentGroup.name]
            for token ∈ tokenList do
 6:             lastEnd ← groupStarts
                silence ← 0
                for segment ∈ segmentGroup do
 9:                 silence ← segment.start − lastEnd
                    lastEnd ← segment.end
                    globalStart ← token.start + groupStarts + silence
12:                 globalEnd ← token.end + groupStarts + silence
                    if segment.start ≤ globalStart ≤ globalEnd ≤ segment.end then
                        token.start ← globalStart
15:                     token.end ← globalEnd
                        addToken(segment, token)
                    end if
18:             end for
            end for
        end for
21: end procedure
```

activate the template usage hyperparameter, and Speak2Subs automatically detects these original subtitles. One potential use of this feature is, as employed in the first part of this thesis, evaluation. With two VTT files having the same structure, Speak2Subs allows us to conduct an evaluation, using the original as a reference.

On the other hand, if we simply want to generate subtitles without the need for them to be tied to a template, Speak2Subs generates them nonetheless, ensuring compliance with standards to guarantee correct readability and display, thereby minimizing human effort in adjusting the final product.

Export to VTT using a template as reference

The algorithm is quite straightforward. It involves loading the original template and iterating through its subtitles. With the timestamps of the words generated by Speak2Subs, forming subtitles that align with the timestamps of the original subtitles is a trivial task. Once the estimated subtitles are formed based on the reference template, they can be exported in VTT format.

Export to VTT without templates

Exporting subtitles in VTT format without a reference template adds a bit more com-

Algorithm 4 Crafting subtitles based on a template reference

```
procedure GenerateFromTemplate(tokenList, templatePath)
    templateSubtitles ← loadTemplate(templatePath)
    predictedSubtitles ← []
    for templateSub ∈ templateSubtitles do
        predictedSubtitle ← ""
        for token ∈ tokenList do                    ▷ For each predicted word
            if templateSub.start ≤ token.end ≤ templateSub.end then
                predictedSubtitle ← predictedSubtitle + token.text
            end if
            predictedSubtitles.append(predictedSubtitle)
        end for
    end for
    return predictedSubtitles
end procedure
```

plexity. Since it's essential to ensure the proper visibility of subtitles, there are several considerations to take into account. For instance, neglecting the display time of subtitles may result in a large number of very short and rapidly changing subtitles. Conversely, grouping as many words as possible in a single subtitle may clutter the screen, making it difficult for viewers to follow the media or read smoothly.

To ensure the production of high-quality subtitles with appropriate display times and smooth subtitle transitions, several considerations have been taken into account, some of which are recommended by the UNE policy framework, as previously mentioned.

It is appropriate to recall the recommendations provided by the UNE 153010:2012 policy framework:

- **4.3 - Number of Lines of Text:** The subtitles should occupy a maximum of two lines of text.
- **4.6 - Number of characters per line:** The maximum character limit per line should be 37.
- **5.1 - Subtitle text exposure speed:** The exposition speed of the subtitle text should follow the rhythm of the original and facilitate comfortable reading. Normally, the maximum number of characters per second would have to be 15 cps (characters per second).

This last recommendation can´t be applied always, because the word timestamps are fixed and innamovible. We can only prolongate the duration of the subtitle if the next one starts after this prolongation, as a best effort.

In addition to those recommendations, two additional considerations are applied: Firstly, whenever possible, subtitles should have a minimum display time of 1 second. Secondly, if

a sentence ends with a period or semicolon, that subtitle concludes, and the next subtitle begins. These two considerations enhance the quality of the subtitles.

Algorithm 5 Crafting subtitles

```
    procedure GENERATE(tokenList)
        predictedSubtitles ← []
        subtitle ← ""
        for token in tokenList do
 5:         if subtitle = "" then
                start ← token.start
            end if
            if EVAL_4.6(subtitle + token.text) or EVAL_4.3(subtitle + NEWLINE()) then
                if EVAL_4.6(subtitle + token.text) then
10:                 subtitle ← subtitle + token.text
                else if EVAL_4.3(subtitle + NEWLINE()newline) then
                    subtitle ← subtitle + NEWLINE() + token.text
                end if
                if EndOfSentence(subtitle) then
15:                 comply, SecondsToComply ← EVAL_5.1(subtitle)
                    if not comply then
                        token.end ← start + SecondsToComply
                    end if
                    predictedSubtitles ← subtitle
20:             end if
            else
                predictedSubtitles ← subtitle
                subtitle ← token.text
                start ← token.start
25:             if EndOfSentence(subtitle) then
                    comply, SecondsToComply ← EVAL_5.1(subtitle)
                    if not comply then
                        token.end ← start + SecondsToComply
                    end if
30:                 predictedSubtitles ← subtitle
                end if
            end if
        end for
        return predictedSubtitles
35: end procedure
```

The algorithm may appear somewhat complex, but with a brief explanation, it can be better understood. Starting with an empty subtitle, for each token (word), a series of checks are performed:

1. If adding the token to the subtitle maintains compliance with the maximum character

limit, it is added.

2. If the previous condition is not met but the token can be added on an additional line, it is added on that line.

3. If, after adding the token through either point 1 or point 2, there is a sentence ending (ending with a period or semicolon), a new subtitle is created. If this subtitle does not meet the minimum duration requirement (5.1), but can do so, its duration is extended until it complies.

4. If neither condition 1 nor condition 2 is met, it means the token belongs to the next subtitle; therefore, the current subtitle is saved. Ultimately, similar to point 3, extending its duration is considered.

In conclusion, the implementation of these conditions when generating subtitles leads to an outcome of higher quality, rendering the subtitles in a more legible way.

4.4.4 Post-processing

With the boom of new LLM (Large Language Model) technologies, we now have increasingly powerful and multidisciplinary tools. As expected, these tools can also be leveraged in post-processing. As observed in the evaluation, models do not capture all details perfectly. Common mistakes include the omission of commas, punctuation marks, or phonetically similar but contextually inappropriate words within a sentence. How can such issues be rectified in the post-processing phase? For instance, in the predicted sentence 'I am going to eat nice with eggs', it is evident that the correct word is 'rice'. Until now, there were not sufficiently potent tools to detect and rectify such errors. However, with the advent of LLMs like GPT or Llama, this possibility becomes available.

In Speak2Subs, a post-processing pilot has been programmed using Llama 2 [17][9], which corrects sentences containing such errors. The only issue is that, being generative tools, they function intermittently—correcting some instances while responding to others without rectifying the intended phrase. Ultimately, the results are somewhat unpredictable, leading to inconsistent post-processing. Implementing a reliable post-processing method with generative LLMs is beyond the scope of this project, but it is worth noting the potential and challenges associated with it.

4.5 How does it work: Subtitle Evaluator

The Speak2Subs programming has been designed with a focus on creating subtitles and their evaluation as entirely independent modules. While they share some methods, this

[9] `https://ai.meta.com/llama-project`

separation allows for the evaluation of files or datasets without relying on runtime-generated information from the generation module. Although such information would have eased the development process, the data obtained from VTT files proves to be sufficient. This part of the package is much lighter than the generation module, but it also has interesting features worth explaining.

There are three phases in the evaluation pipeline: Detecting VTT files, calculating metrics, and exporting to a final Excel. Figure 4.5 is a good overview of the evaluator pipeline.

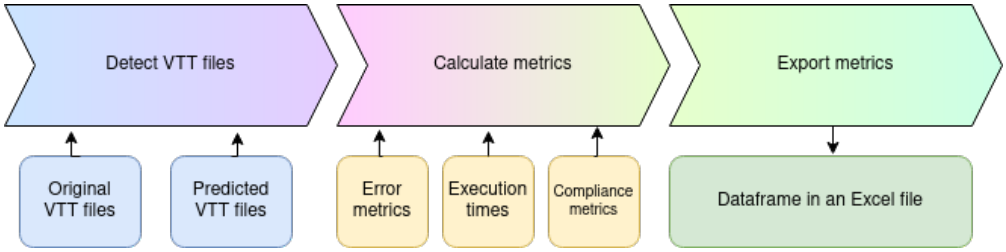

Figure 4.5: Evaluation pipeline

4.5.1 Detect VTT files

Detecting the original and generated VTT files is a straightforward task, but it is crucial to maintain the folder and file structure generated in the previous step. This means that the original VTT files should remain in the dataset folder, and the generated VTT files should stay in the folders created within the results path, where each ASR system has its dedicated folder. While this approach may seem less flexible, the trade-off is a straightforward automation, resulting in easy utilization from the command line interface (CLI) or scripts. In any case, individual evaluations can always be performed without considering specific names or paths.

4.5.2 Calculate metrics

The most crucial aspect of the evaluation is the calculation of metrics. It is important to note that there are three types of metrics: error metrics, compliance metrics, and runtime metrics. Each metric is calculated differently, so let's go into the specific calculation process for each one separately.

Error metrics

These metrics are computed by comparing each file, subtitle by subtitle. Therefore, it is a prerequisite that both files have the same number of entries to avoid calculation errors. If the prediction is generated using Speak2Subs and the original VTT is used as a template,

there is no issue. After loading both templates, the subtitles are iterated over, and error metrics are calculated.

Without delving into the specific list of metrics that compose the error metrics (which can be found in their respective section), it is essential to emphasize the four types of error metrics:

1. Non-normalized metrics
2. Normalized metrics
3. Mismatch metrics
4. NER metrics

The first two are calculated using the 'jiwer' package (https://jitsi.github.io/jiwer/). However, to obtain normalized metrics, a preprocessing step is performed, wherein uppercase letters, punctuation marks, and other symbols are removed from both subtitles. Mismatch and NER metrics are calculated using a custom algorithm.

The mismatch algorith is pretty simple, for each pair of subtitles, it looks if the first word of some of them two matchs the last word of the previous subtitle of the pair. Before comparing, the words are normalized. The algorithms keeps the count, and it calculates the final rate of mismatch words.

Algorithm 6 Mismatch algorithm

```
procedure MismatchRate(referenceSubs, predictedSubs)
    mislocated ← 0
    total ← len(referenceSubs)
    for i in 1 to total − 1 do
        LastRefWord ← NORM(referenceSubs[i − 1][−1]
        CurrentPredWord ← NORM(predictedSubs[i][0]
        if LastRefWord ≠ CurrentPredWord then
            mislocated ← mislocated + 1
        end if
        CurrentRefWord ← NORM(referenceSubs[i][0]
        LastPredWord ← NORM(predictedSubs[i − 1][−1]
        if CurrentRefWord ≠ LastPredWord then
            mislocated ← mislocated + 1
        end if
    end for
    return mislocated/total
end procedure
```

The NER metric, as it is explained in the metrics section, measures the impact of edition mistakes (same word but with differences) and recognition mistakes (different word). So the algorithm is pretty straightforward:

Algorithm 7 NER metric algorithm

```
procedure CalculateNER(referenceSubs, predictedSubs)
    N ← 0
    E ← 0
    R ← 0
    for (refSub, predSub) in (referenceSubs, predictedSubs) do
        for predWord in predSub do
            if predWord not in CurrentPredWord then
                if NORM(predWord) in NORM(CurrentPredWord) then
                    E ← E + 1                                   ▷ Word edited
                end if
            else if predWord ∉ CurrentPredWord then
                if NORM(predWord) ∉ NORM(CurrentPredWord) then
                    R ← R + 1                                   ▷ Word not recognited
                end if
            end if
            N ← N + 1
        end for
    end for
    return N + E + R/N
end procedure
```

Execution times metrics

When generating subtitles, the runtime durations employed by each ASR model for every audio file are recorded. This process ensures a comprehensive record of the processing speed associated with each model and measures the impact of hyperparameter selection on said model speed. These records are stored in the results folder, in a JSON file, and are organized by the audio file name as the key, with all used ASR model times as values.

During the evaluation phase, Speak2Subs reads and parses this file, extracting the data to integrate it into the metrics Excel spreadsheet as an additional column.

Compliance metrics

Compliance metrics show a distinctive characteristic, which is, by their inherent nature, that they can be obtained without the reliance on a reference VTT file. Chapter 4 extensively discusses the application of compliance policies to generated subtitles in the absence of a reference template. In the evaluation phase presented here, given that the generated subtitles faithfully replicate the structure of the original template, it is evident that this aspect does not warrant further improvement.

Nevertheless, quantifying these metrics is straightforward and shows the extent of human interventions that may be required for refinement. Pseudo-code doesn't seem neccesary, a brief description is enough to explain how the compliance metrics are obtained.

1. 4.3 - The subtitles should occupy a maximum of two lines of text. For each subtitle, if the text has more than one newline character, the counter adds one.

2. 4.6 - The maximum character limit per line should be 37. For each subtitle, if the text has more than 37 characters, the counter adds one.

3. 5.1 - The maximum number of characters per second would have to be 15 cps. If the division of the number of characters by the duration of the subtitle is greater than 15, the counter adds one.

In the three of them, returns the counter divided by the total number of subtitles.

4.6 Development costs

This project has been developed using a rudimentary computer setup, without dedicated graphics cards or any elements incurring extraordinary costs. Therefore, the only cost factor to be considered would be the hypothetical cost of a developer working at an hourly rate of X. Approximately 250 hours were invested in software development, with an additional 50 hours allocated to the composition of this project thesis. If this thesis is considered the project's documentation, it could be estimated that the hypothetical economic cost to produce this project has been in the vicinity of 300*X, where X represents the gross hourly wage of a software developer, scientist, or data engineer.

4.7 Development conclusions

The package developed for this thesis possesses a crucial characteristic: it operates as a black box. With a single command, high-quality subtitles can be obtained from a multitude of files. This feature stands as a significant advantage for potential users seeking out-of-the-box functioning tools. In conclusion, it can be stated that the initial objective pertaining to the package has been achieved.

Chapter 5

Analysis of results

In this section, the evaluation results are presented in detail. It is not enough to display raw data; the goal is to derive meaningful conclusions by interpreting the results accurately. Given that the comprehensive results have a substantial amount of information, this section presents the global averages across all assets from the five datasets. Complete tables are referenced and stored on the GitHub package repository.

The aggregated averages serve as a concise overview, offering insights into the overall performance of the ASR models across various datasets. The detailed tables at the conclusion of this document provide a more granular view of the results for those seeking a deeper understanding of the evaluation outcomes.

What are the implications of using ASR models without the post-processing or pre-processing that Speak2Subs employs? Does employing Speak2Subs actually offer any enhancement compared to utilizing the raw output from these models? To address these inquiries, there is also a comparative analysis of subtitle quality, contrasting the raw output from the most effective ASR models with the output generated by Speak2Subs. This comparison does not assess accuracy in terms of transcription, as no template is used, but evaluates the adherence to UNE standards for subtitles derived directly from the model, and contrasts it with the subtitles produced by our package.

5.1 Evaluating results using the configurations

To conduct evaluation, the four configurations described in 3.4 are employed. Based on the outcomes from each configuration, conclusions are drawn regarding the effectiveness of diverse models and preprocessing techniques.

Full metrics can be download in the GitHub repository. [1]

[1] `https://github.com/JulioFresneda/Speak2Subs/blob/main/TestMetrics/`

5.1.1 Configuration 1 - Default

The default hyperparameters represent the theoretically optimal configuration. This configuration incorporates VAD, segmentation to alleviate computational effort, and grouping segments to preserve contextual information. Theoretically, this configuration is the one that should get better results.

The results are pointed out in Table 5.1. Lower is better, and the best result is written in bold.

Table 5.1: Configuration 1 - Default hyperparameters. Except exec time, each metric is measured in percentage (%).

Metric	nemo	seamless	vosk	whisper	whisperx
wer	39	66	41	28	**21**
mer	37	59	39	26	**20**
wil	52	75	56	34	**28**
wip*	48	25	44	66	**72**
wwer	30	48	33	19	**15**
wmer	28	43	31	17	**15**
nwer	27	60	28	22	**14**
nmer	25	53	25	20	**13**
nwil	33	68	35	24	**16**
nwip*	67	32	65	76	**84**
nwwer	18	41	19	13	**8**
nwmer	16	37	18	12	**8**
mislocation rate	32	25	**19**	35	**19**
ner	85	65	82	87	**89**
exec time (s)	35	444	**28**	455	331

**In WIP and NWIP metric, a higher value indicates better performance.*

With the default hyperparameters, there is a clear winner: WhisperX. This outcome is not surprising, considering that WhisperX has established itself as a robust ASR system with impressive results, even achieving 1st place at the Ego4d transcription challenge. It excels in transcription accuracy. However, it's worth noting that WhisperX lags behind in terms of execution speed, where both NeMo and Vosk demonstrate significantly faster performance.

This observation highlights the trade-off between accuracy and speed among the evaluated ASR tools. While WhisperX excels in accuracy, users may need to consider the specific requirements of their applications and whether faster processing times are a critical factor in their decision-making process.

It's noteworthy that Whisper closely follows WhisperX in performance. Additionally, the observation that WhisperX remains faster than Whisper is intriguing, considering that WhisperX applies Voice Activity Detection (VAD) within its model. This implies that, with these default hyperparameters, VAD is applied twice in the case of WhisperX. The fact

that WhisperX maintains its speed advantage despite this double application of VAD is an interesting aspect to consider.

This information underscores the efficiency of WhisperX and its ability to handle the additional computational load introduced by dual VAD processing, which is crucial for applications where both accuracy and speed are essential considerations.

5.1.2 Configuration 2 - No VAD

Without VAD, the models will receive not only speech as input but also segments of silence. Let's explore the consequences of deactivating it.

The results are pointed out in Table 5.2. Lower is better, and the best result is written in bold.

Table 5.2: Configuration 2 - No VAD applied. Except exec time, each metric is measured in percentage (%).

Metric	nemo	seamless	vosk	whisper	whisperx
wer	35	66	38	**19**	21
mer	34	59	36	**18**	20
wil	51	74	54	**27**	29
wip*	49	26	46	**73**	71
wwer	29	46	32	**15**	**15**
wmer	29	42	30	**14**	15
nwer	20	59	23	**12**	14
nmer	19	53	21	**11**	13
nwil	26	65	30	**14**	16
nwip*	74	35	70	**86**	84
nwwer	14	39	17	**8**	**8**
nwmer	13	35	16	**7**	8
mislocation rate	**6**	30	12	8	18
ner	86	67	83	**91**	90
exec time (s)	40	395	**32**	691	346

**In WIP and NWIP metric, a higher value indicates better performance.*

In general, with VAD applied, the error results have improved. The significant beneficiary is Whisper, showing a substantial improvement, reaching an impressive 8 NWWER! The other models have improved slightly, but nothing remarkable. WhisperX maintains almost the same results, which is logical since it already applies VAD natively. The downside of deactivating VAD is the execution speed. The following subsections will delve into conclusions regarding this matter.

5.1.3 Configuration 3 - Sentences

In this configuration, the goal is to test the importance of context in the speech-to-text task. The models process independent phrases without having information about the rest of the audio, requiring them to leverage those very few seconds to achieve satisfactory results.

The results are pointed out in Table 5.3. Lower is better, and the best result is written in bold.

Table 5.3: Configuration 3 - Sentences, grouping disabled. Except exec time, each metric is measured in percentage (%).

Metric	nemo	seamless	vosk	whisper	whisperx
wer	34	38	38	**20**	24
mer	32	34	36	**20**	23
wil	50	48	54	**31**	36
wip*	50	52	46	**69**	64
wwer	29	28	32	**17**	20
wmer	28	26	31	**16**	19
nwer	18	29	23	**11**	14
nmer	17	26	21	**10**	13
nwil	25	34	31	**14**	17
nwip*	75	66	69	**86**	83
nwwer	13	19	17	**7**	9
nwmer	13	17	16	**7**	8
mislocation rate	**5**	23	11	6	13
ner	85	80	83	**88**	85
exec time (s)	**39**	480	68	1331	1247

**In WIP and NWIP metric, a higher value indicates better performance.*

Whisper remains the winner. The results are almost identical to those of Configuration 2. There is a slight negative variation in the non-normalized metrics and a slight positive variation in the normalized metrics. This could indicate that the models use context for proper punctuation, such as periods, commas, and other punctuation marks. However, the variation is so small that we cannot claim significantly relevant differences.

5.1.4 Configuration 4 - Raw audio

In this configuration, the objective is to assess whether the model exhibits sufficient capability to transcribe audio of substantial length effectively or if it is necessary to segment the audio into batches.

This configuration shows the incapacity of three out of the five models to process such lengthy audio inputs. Whether due to computational limitations or inherent model constraints, Seamless, WhisperX, and NeMo have proven unable to provide results without experiencing crashes. So, in this configuration, only Vosk and Whisper are used.

The results are pointed out in Table 5.4. Lower is better, and the best result is written in bold.

Table 5.4: Configuration 4 - Raw audio, no VAD and no segmentation. Except exec time, each metric is measured in percentage (%).

Metric	vosk	whisper
wer	37	**19**
mer	35	**18**
wil	53	**26**
wip*	47	**74**
wwer	32	**14**
wmer	30	**14**
nwer	22	**12**
nmer	21	**12**
nwil	29	**14**
nwip*	71	**86**
nwwer	16	**7**
nwmer	15	**7**
mislocation rate	12	**8**
ner	84	**92**
exec time (s)	**21**	664

**In WIP and NWIP metric, a higher value indicates better performance.*

Only Whisper and Vosk have successfully transcribed the lengthy audio inputs. Whisper excels in subtitle precision, and Vosk y execution time. From this, we can infer that if, for any reason, segmentation of the audio for transcription is not feasible, the choice would likely be one of these two models. And depending of the task, would be suited a precise model like Whisper or a fast one like Vosk.

5.2 Metric comparatives

It's worth zooming in on specific metrics as valuable insights can be drawn. We can infer some noteworthy conclusions.

5.2.1 Error rate comparative - WWER

The WWER comparative can be seen in Table 5.5. As always, lower is better, and the best result is remarked in bold.

Table 5.5: WWER comparative (%).

Configuration	nemo	seamless	vosk	whisper	whisperx
Conf 1 - Default	30	48	33	19	**15**
Conf 2 - No VAD	29	46	32	**15**	**15**
Conf 3 - Sentences	29	28	32	**17**	20
Conf 4 - Raw audio	N/A	N/A	32	**14**	N/A

NeMo and Vosk exhibit remarkable stability; they are indifferent to processing information with or without VAD and the duration of the segments. The close similarity in results between NeMo and Vosk speaks either strongly against NeMo or in favor of Vosk, considering that NeMo theoretically employs more advanced technologies and has a heavier model.

Additionally, there is an extraordinary improvement in the case of Seamless when dealing directly with sentences. This makes sense since Seamless doesn't generate word-level timestamps, and Speak2Subs provides a very rudimentary estimation. The shorter the segments, the more accurate Seamless's estimation becomes, leading to better results.

Whisper and WhisperX effectively leverage context, as both show a decline in performance in Configuration 3, and the best result is Whisper, in config 4. Interestingly, without VAD, Whisper experiences a four-point drop, equalizing with WhisperX.

5.2.2 Error rate comparative - NWWER

The NWWER comparative can be seen in Table 5.6. As always, lower is better, and the best result is remarked in bold.

Table 5.6: NWWER comparative (%).

Configuration	nemo	seamless	vosk	whisper	whisperx
Conf 1 - Default	18	41	19	13	**8**
Conf 2 - No VAD	14	39	17	**8**	**8**
Conf 3 - Sentences	13	19	17	**7**	9
Conf 4 - Raw audio	N/A	N/A	16	**7**	N/A

If we look at the normalized metric, there is a substantial overall improvement, which is evident as we eliminate errors related to capitalization and punctuation. There are no major conclusions that we haven't already obtained in the previous section.

5.2.3 Mislocation rate comparative

The mislocation rate comparative can be seen in Table 5.7. As always, lower is better, and the best result is remarked in bold.

Table 5.7: Mislocation rate comparative, in seconds.

Configuration	nemo	seamless	vosk	whisper	whisperx
Conf 1 - Default	32	25	**19**	35	**19**
Conf 2 - No VAD	**6**	30	12	8	18
Conf 3 - Sentences	**5**	23	11	6	13
Conf 4 - Raw audio	N/A	N/A	12	**8**	N/A

The ratio of words with incorrect timestamps is very useful for measuring the precision of the models. It's important to note that we are assuming the timestamps of the original subtitles are accurate and correct. In general, there is a significant improvement when VAD or grouping is disabled. In both scenarios, it makes sense:

1. With VAD disabled, there are far fewer audio cuts, specifically one every 30 seconds, with perfectly placed timestamps. 2. With VAD enabled but without grouping, we still have these segments, but without grouping, there's no need to calculate periods of silence between sentences. If there are small errors in these calculations, which accumulate, they can lead to words with imprecise timestamps.

5.2.4 Execution times comparative

In table 5.8 a raw execution time comparative can be seen. As always, lower is better, and the best result is remarked in bold. In table 5.9 we can see the execution time ratios, using Vosk with Conf 1 as reference.

Table 5.8: Execution time comparative, in seconds.

Configuration	nemo	seamless	vosk	whisper	whisperx
Conf 1 - Default	35	444	**28**	455	331
Conf 2 - No VAD	40	395	**32**	691	346
Conf 3 - Sentences	**39**	480	68	1331	1247
Conf 4 - Raw audio	N/A	N/A	**21**	664	N/A

Table 5.9: Execution time ratios, Vosk in Conf 1 as reference.

Configuration	nemo	seamless	vosk	whisper	whisperx
Conf 1 - Default	x1.25	x15.85	x1	x16.25	x11.82
Conf 2 - No VAD	x1.42	x14.10	x1.14	x12.43	x12.35
Conf 3 - Sentences	x1.39	x17.14	x2.42	**x47.53**	x44.53
Conf 4 - Raw audio	N/A	N/A	x0.75	x23.71	N/A

The information on execution times is crucial. If speed is prioritized, it's evident that Vosk and NeMo should be the winning choices. They improve approximately by a factor of

10 compared to the competition, regardless of the audio length. For Whisper and WhisperX, the length of audio matters significantly: their execution times triple when used sentence by sentence.

5.2.5 Memory consumption comparative

It is mandatory to have an idea of what amount of memory the user has to have in his system. In table 5.10, it is reflected the average RAM consumption for the transcription jobs.

Table 5.10: Memory consumption, in Gb.

Model	Average RAM consumption
Whisper	6.6
WhisperX	3.6
Seamless	9.4
NeMo	0.5
Vosk	**0.4**

5.3 Evaluation conclusions

Several conclusions can be drawn from the available data. What is the best model? It depends.

If you aim for excellence in subtitles, regardless of execution time, and want to minimize human intervention, the best option is WhisperX with Configuration 2. If we look at WWER, Whisper and WhisperX share the best results. Since WhisperX is faster, it could be considered the top choice. Although Whisper has a slight advantage in normalized metrics, since the goal is to minimize human correction, it's essential for the model to interpret capitalization and punctuation correctly. On the other hand, sentence-level results don't improve enough when grouping to offset the substantial additional time used by Whisper and WhisperX. Whisper with configuration 4 is not recommended because of the computational cost. It is highly recommended to divide the audio into batches for security reasons.

If, however, speed is the priority, NeMo and Vosk would be the ideal choice. Vosk works out of the box, while with NeMo, it might be necessary to evaluate different submodels; some could improve results.

Regarding hyperparameter selection, VAD enhances speed by eliminating silences; whether it compensates for the loss of quality is a decision for the end user. Sentences-level should not be recommended to use, althought maybe is a good indicator of how the models could perform real-time subtitulation.

Finally, it should be noted that the best result of the evaluation is for Whisper with 'mda_3' audio, with an astounding score of 7% WWER and 1.5% NWWER. That's almost

a perfect score!

5.4 Evaluating subtitles without a template reference

The evaluation conducted in the preceding sections underscores the robust capabilities of the models in generating subtitles, relying on original and presumably flawless subtitles as a reference. However, when utilizing Speak2Subs to generate new subtitles, the norm is not to have reference subtitles available.

As detailed in other sections of the document, Speak2Subs offers both template-based and template-free modalities. Does this distinction yield differences in quality? The answer is no.

The transcription remains consistent, whether a template is employed or not. The sole disparity lies in the organization of sentences. In fact, generating subtitles without a template presents an advantage: adherence to UNE 153010:2012 compliance recommendations. Given Speak2Subs' freedom to construct sentences, there is greater flexibility in subtitle creation. This flexibility mitigates issues such as truncated sentences or subtitle continuations after punctuation marks. Let us examine some examples.

Let's watch 'mda_1.mp4' video with the original subtitles:

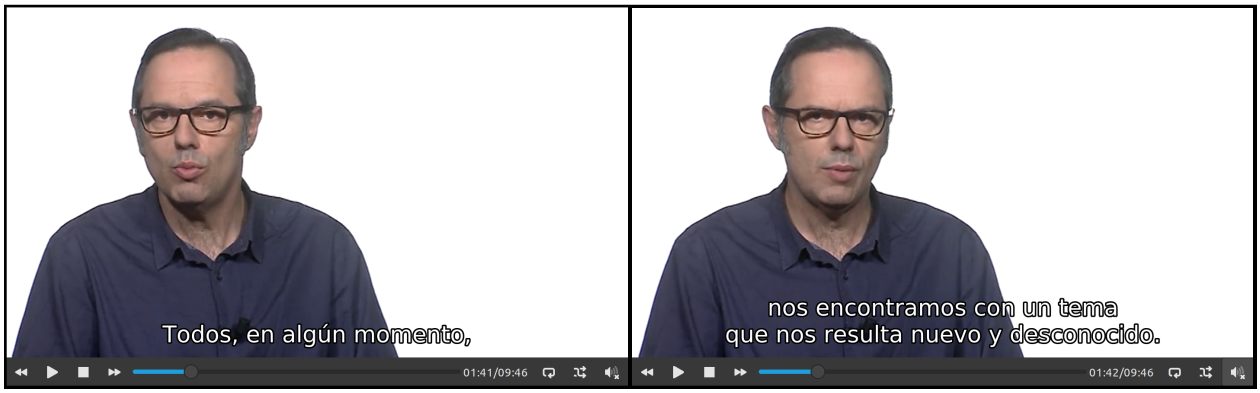

Figure 5.1: mda_1.mp4 video, at 1:41 with original subtitles

In this video frames, the original subtitles exhibit a certain issue. While it may not seem significant, for someone who needs to read subtitles for extended periods, it can lead to mental fatigue more rapidly.

The problem lies in the subtitles being too brief, appearing within a very short timeframe. If this pattern persists, the viewer is required to read constantly short and quickly appearing subtitles. In essence, the original subtitles do not align with the compliance policies of the UNE 153010:2012.

Let's now examine the same frames, generated by Speak2Subs:

With the generated subtitles, in the first frame, an effort is made to include the maximum amount of text within the subtitle while adhering to UNE 153010:2012 policies 4.3 and

Figure 5.2: mda_1.mp4 video, at 1:41 with generated subtitles

4.6. This approach allows the viewer to read the subtitle more comfortably. In the second frame, upon completing a sentence with a period, the subtitle is truncated, reserving the continuation for the next subtitle. This strategy enhances the overall readability and conforms to compliance policies, addressing the potential mental fatigue associated with rapidly appearing and short subtitles.

This is an example, but to draw conclusions, it is necessary to scrutinize the metrics. To clarify the table, the UNE 153010:2012 4.3 is called "Lines", the 4.6 is called "Line width" and the 5.1 is called "Speed".

The table reflects percentages. Every track of the subtitles are evaluated, and 100% mean that all of the subtitle tracks comply with the policy. For example, the original subtitle has an 82% in Line width. That means that about 28 of 153 subtitle tracks (18%) have more characters than recommended.

Table 5.11: Compliance comparative (%)

Subtitles	Lines	Line width	Speed / Total subtitle tracks	
Original	100	82	40	153
Generated	100	100	42	145

Upon examining the metrics, it becomes evident that the generated subtitles align perfectly the UNE 153010:2012 policies, whereas, in the original subtitles, adherence to these policies is not a priority.

It is an example, but it can be extrapolated to the rest of the datasets. It is not neccesary to evaluate all datasets because we will have similar results.

In conclusion, by not using a template, we maintain the same error rate and, additionally, achieve a perfect percentage of compliance in 4.3 and 4.6, and a best effort in 5.1, with UNE 153010:2012 recommendations regarding subtitles.

5.5 WhisperX vs Speak2Subs: A comparative analysis of subtitle quality

In previous sections, the high quality of the tested models has been highlighted. The possibility of evaluating the UNE 153010:2012 criteria has also been discussed, underscoring their significant importance.

However, there remains an unresolved question. The reader might wonder whether it is truly worthwhile to use Speak2Subs instead of directly employing an ASR model. Take WhisperX, for instance, which not only performs speech-to-text to obtain the transcription of each word with its associated timestamp but also applies native Voice Activity Detection, resulting in batches of sentences with timestamps. The reader could think that perhaps it is simpler to use WhisperX, which has proven to be exceptional, and use these sentences as subtitles. And indeed, this is a practice that is employed. With such high precision results, what advantage does using Speak2Subs offer over WhisperX?

Of course, a detailed response can be provided, but ideally, the answer should be quantitative: In this section, the compliance with UNE 153010:2012 criteria of WhisperX's native output of sentences is compared with the output offered by Speak2Subs. This allows for an analysis of whether there is a significant difference in subtitle quality.

WhisperX is used because of the excellent results and being able to generate "subtitles" or complete sentences, not only words, which is a feature that not all ASR models have.

The comparison is shown in the following table. The percentage of compliance with standards 4.3 (number of lines per subtitle), 4.6 (number of characters per line), and 5.1 (maximum subtitle speed) has been compared, on a scale of 0-1, where 1 represents perfect compliance and 0 represents no compliance.

Table 5.12: WhisperX vs Speak2Subs.

Dataset	UNE 4.3	UNE 4.6	UNE 5.1
mda			
WhisperX	1.0	0.09	0.06
Speak2Subs	1.0	**1.0**	**0.22**
mat			
WhisperX	1.0	0.13	0.14
Speak2Subs	1.0	**1.0**	**0.28**
cdp			
WhisperX	1.0	0.14	0.02
Speak2Subs	1.0	**1.0**	**0.2**
atc			
WhisperX	1.0	0.11	0.06
Speak2Subs	1.0	**1.0**	**0.26**
ddl			
WhisperX	1.0	0.09	0.06
Speak2Subs	1.0	**1.0**	**0.3**

From the previous table, quite interesting conclusions can be drawn. Looking at rule 4.3, which states that the maximum number of lines per subtitle track is 3, both WhisperX and Speak2Subs meet this criterion perfectly. In the case of WhisperX, this is logical, as by default the model does not insert line breaks in overly long sentences. Indeed, WhisperX's ability to meet this criterion is also a drawback for meeting 4.6, as it significantly exceeds the maximum number of characters per subtitle. This can be problematic for viewers of the subtitled multimedia file, as the subtitles will saturate the screen with overly long phrases. This might be overlooked if the results for criterion 5.1 were good, as it would mean that viewers have enough time to read such long subtitles.

However, this is not the case. As can be seen, the results are poor, which is a double defect: on one hand, we have overly long subtitles, and on the other, little time is given for reading them. In the case of Speak2Subs, both criteria 4.3 and 4.6 are always met 100 percent. This is not a coincidence, simply Speak2Subs applies a post-processing that forces subtitles to meet these criteria algorithmically, by design they will always be met 100 percent.

This quality comes at a price, and that is criterion 5.1. Speak2Subs forces subtitles not to be too long, and if the speaker is too fast, inevitably the subtitle tracks must follow at that same speed, to maintain synchronization with the audio. In other words, Speak2Subs sacrifices criterion 5.1 to achieve perfect scores in 4.3 and 4.6.

Why, even sacrificing the speed criterion, does it achieve better results than WhisperX,

which does not make this sacrifice? Because Speak2Subs uses additional specific post-processing for this criterion. Among other things, it detects occasions when a subtitle track is preceded by silence, and takes advantage of this to extend the duration of that subtitle, as it will not overlap with the next. In this way, Speak2Subs can give the viewer a breather and allow them to read the subtitle until the next one arrives.

In conclusion, while the accuracy of WhisperX's and other models' transcriptions is excellent, their out-of-the-box use to obtain subtitles is not advisable without adequate post-processing, as they do not meet minimum criteria for proper readability and viewer comfort. Therefore, if the task is specifically to obtain subtitles, it is recommended to use Speak2Subs over unprocessed models like WhisperX.

Chapter 6

Conclusions and final thoughts

If we employ out-of-the-box Automatic Speech Recognition (ASR) models without any modifications to the outcomes, it is evident that the transcription is unsuitable for use as high-quality subtitles. This is due, among other reasons, to the lack of consideration for certain modifications in the output to tailor it to the subtitle format. Among these potential modifications are those suggested by the UNE 153010:2012 standards, some of which are implemented by Speak2Subs.

6.1 Speak2Subs VS State-of-the-art

As can be seen in 2.1, there are particularly interesting examples of the use of ASR for subtitling, which highlights that Speak2Subs is not the only project attempting to subtitle using ASR models. However, there are some differences between Speak2Subs and the models described in 2 citations. For example, the article [13] focuses on subtitling for TV programs, which typically have default subtitles of medium to low quality. The model in this article utilizes these existing subtitles, whereas Speak2Subs does not have this capability. Therefore, for this specific task, the model proposed in the article is more suitable than Speak2Subs.

Evidently, Speak2Subs does not share the same objective as the model in the article; Speak2Subs aims for a more global application, is much more versatile, and does not depend on reference subtitles to achieve exceptional results.

The article [15] adopts an approach similar to Speak2Subs, and there are phases that are alike: the extraction of audio from the video, division into chunks, application of the ASR model to obtain tokens with their timestamps, and the creation of a subtitle file suitable for addition in any media player. While the results are commendable, and it is a robust model, there are several features that differentiate Speak2Subs from this work. These include Voice Activity Detection (VAD), the selection of the ASR model to be used, post-processing adjustments in line with UNE 153010:2012 standards, and some additional hyperparameters.

In the article[4], the approach is indeed intriguing, and its model operates in a manner

somewhat similar to Voice Activity Detection. Instead of dividing the entire audio at silences as VAD does, it marks the audio by annotating where there are silences, that is, subtitle breaks. This presents an interesting alternative to the use of VAD and UNE 153010:2012 criteria. However, as a trade-off, it slightly reduces the precision of the transcriptions, by about 2-3

While this examples shares a similar objective with Speak2Subs, it is evident that this examples, among many others, attempts to address the problem in a considerably lighter, less comprehensive, and less thorough manner compared to Speak2Subs.

The article described in 2.1.2 is very interesting too. In the case of Speak2Subs, subtitle compression is not utilized, instead it tries to adhere to specific criteria that yield very good results in terms of readability, without the need for information loss or compression.

As we have seen in numerous papers and articles, Speak2Subs is not the first project to attempt to use ASR models to generate high-quality subtitles. Without intending to overstate the capabilities of Speak2Subs, a one-to-one comparison with the various models and projects presented in 2.1 section reveals that either Speak2Subs is compatible with more generic tasks than those for which the comparative model is designed, or the comparative models lack the depth, customizability, and adaptability offered by Speak2Subs. The optional use of Voice Activity Detection, segmentation grouping, the choice among five different ASR models, and post-processing according to UNE 153010:2012 criteria are some of the features that Speak2Subs offers, which have not been reflected in any other paper or project we have reviewed.

6.2 What has been achieved

In conclusion, the project's objectives have been successfully achieved on multiple fronts.

Firstly, a comprehensive comparison of state-of-the-art ASR models has been conducted, yielding clear and insightful conclusions. The performance of each model, its strengths, and weaknesses has been thoroughly examined. Depending on the end user's specific requirements, this comparative analysis provides a clear understanding of which models are better suited for their particular use case.

Simultaneously, the development of a package capable of generating subtitles with outstanding quality has been accomplished. Unlike the use of unmodified ASR models, Speak2Subs does apply certain adjustments to the output, ensuring that the result is suitable for use with minimal subsequent processing. This package operates as a black box, allowing users to effortlessly obtain perfectly synchronized VTT format subtitles with minimal errors, without necessitating specialized knowledge. The package's installation is straightforward and does not demand a high-powered computing environment. In summary, the initial project objectives have been satisfactorily resolved.

6.3 What can be improved

While the quality of the results is satisfactory, there is always room for improvement. As explained in earlier sections, post-processing 4.4.4 with a Language Model (LLM) layer applied to each subtitle could notably enhance the final outcomes.

Another potential enhancement, though contemplated but not implemented, is a voting system. For each subtitle, all five models could work concurrently, predicting results and selecting each word based on majority votes. This idea has not been executed due to computational constraints, but it remains an open avenue that could be explored in the future.

Lastly, an improvement for the future involves the incorporation of new ASR models. Given that Speak2Subs utilizes containers, integrating a new ASR model simply requires adding a new image without the need for substantial code modifications. There is always room for improvement, and as technology advances, new ideas emerge.

Regarding the UNE 153010:2012 standards, it is important to note that many of the recommendations fall outside the scope of Speak2Subs due to its inherent nature. Visual recommendations, such as contrast and color selection, for example, are obviously not applicable to this project. However, some recommendations, such as indicating the speaker in a dialogue involving multiple people, are feasible for implementation. Perhaps in the future, this package might be capable of identifying different speakers, thereby making the application of such recommendations viable.

Bibliography

[1] Max Bain et al. *WhisperX: Time-Accurate Speech Transcription of Long-Form Audio*. July 11, 2023. DOI: 10.48550/arXiv.2303.00747. arXiv: 2303.00747[cs,eess]. URL: http://arxiv.org/abs/2303.00747.

[2] Aishwarya Bakale et al. "Video Subtitle Generation". In: ().

[3] William Chan et al. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621.

[4] Maria Federico and Marco Furini. "An automatic caption alignment mechanism for off-the-shelf speech recognition technologies". In: *Multimedia Tools and Applications* 72.1 (Sept. 1, 2014), pp. 21–40. ISSN: 1573-7721. DOI: 10.1007/s11042-012-1318-3. URL: https://doi.org/10.1007/s11042-012-1318-3.

[5] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[6] J. Howard and S. Gugger. *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated, 2020. ISBN: 9781492045526. URL: https://books.google.no/books?id=xd6LxgEACAAJ.

[7] Matthew J. Hunt. "Weighted WER for Speech Recognition". In: *Unknown* (1990).

[8] Shuai Li et al. "Recent Advances in End-to-End Automatic Speech Recognition". In: *arXiv preprint arXiv:2111.01690* (2021). URL: https://ar5iv.org/html/2111.01690.

[9] Jingzhou Liu et al. *An Approach to Improve Robustness of NLP Systems against ASR Errors*. 2019. arXiv: 1909.09577 [cs.CL].

[10] Andrew Morris, Viktoria Maier, and Phil Green. *From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition*. Oct. 4, 2004. DOI: 10.21437/Interspeech.2004-668.

[11] Author Names. "Adapting End-to-End Speech Recognition for Readable Subtitles". In: *arXiv preprint arXiv:2005.12143* (2020). URL: https://arxiv.org/abs/2005.12143.

[12] Jakob Poncelet and Hugo Van hamme. *Learning to Jointly Transcribe and Subtitle for End-to-End Spontaneous Speech Recognition*. 2022. arXiv: 2210.07771 [cs.CL].

[13] Jakob Poncelet and Hugo Van Hamme. "Learning to Jointly Transcribe and Subtitle for End-To-End Spontaneous Speech Recognition". In: *2022 IEEE Spoken Language Technology Workshop (SLT)*. 2023, pp. 182–189. DOI: 10.1109/SLT54892.2023.10022420.

[14] Alec Radford et al. *Robust Speech Recognition via Large-Scale Weak Supervision*. Dec. 6, 2022. DOI: 10.48550/arXiv.2212.04356. arXiv: 2212.04356[cs,eess]. URL: http://arxiv.org/abs/2212.04356 (visited on 12/09/2023).

[15] Aditya Ramani et al. "Automatic Subtitle Generation for Videos". In: *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. 2020, pp. 132–135. DOI: 10.1109/ICACCS48705.2020.9074180.

[16] *Seamless: Multilingual Expressive and Streaming Speech Translation | Research - AI at Meta*. URL: https://ai.meta.com/research/publications/seamless-multilingual-expressive-and-streaming-speech-translation/ (visited on 12/09/2023).

[17] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. July 19, 2023. DOI: 10.48550/arXiv.2307.09288. arXiv: 2307.09288[cs]. URL: http://arxiv.org/abs/2307.09288 (visited on 12/09/2023).

[18] *UNE 153010:2012*. https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0049426. Subtitling for Deaf and Hard-of-Hearing People. 2012.

[19] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.