Automatically Converting Speech to Song

Master's Thesis

Student: C.F. de Mari
Student Number: 5696348

First examiner: Prof. dr. R.C. Veltkamp
Second examiner: Dr. Anja Volk

Game and Media Technology
Utrecht University

Netherlands, June 2022

Preface

This thesis is the final project for the Master Game and Media Technology at the University of Utrecht. In it, I created a pipeline that automated every step necessary to create a cover of a song (using concatenated clips of one talking speaker to align with the lyrics), using only videos of the speaker and the song audio as input. This pipeline and the results it produces are the topic of the research performed in this thesis. Throughout the process, weekly meetings with my main supervisor Prof. Dr. Remco C. Veltkamp were crucial in supporting the creation of this thesis, and for that I express my gratitude.

1 Introduction

Cover versions of songs have been created for many decades. The history of records getting covered goes back to at least the 1950's [1], but undoubtedly songs were being covered long before then. Cover songs are defined by the fact that a different artist is singing the lyrics, but it is not uncommon for the covers to feature changes in the instrumentation and genre as well. Creating a cover in the early days required musicians to play and sing the original tracks, which is a high requirement to have for the average person compared to modern day musical editing. Nowadays the minimum requirement for working on music is a lot lower, where even a person without a musical background only needs access to a computer with the right software to start playing around with music. The process of spreading ones musical creations has also been simplified through modern technology. The arrival of the internet allows for anyone to upload their own music, where the rest of the world can view it. As a result of the combination of these two factors, the internet is currently being flooded with ten of thousand of new tracks every day [2]. So despite the relative ease of creating and uploading music these days, artists do not have it easy.

Over the years there have been YouTube channels with viral music cover videos, created using innovative methods. The channel Schmoyoho [3] has created viral song videos that attracted more than tens of millions of views. As opposed to covers, where a song is the inspiration and the artist gets replaced, Schmoyoho took one or multiple quotes from a person's video and turned it into a song sang by them.

Another channel named Baracksdubs [4] specialized in taking footage from Barack Obama, and cutting and pasting clips of him into a song to give the impression that it was Barack Obama singing, with his most popular video "Barack Obama Singing Call Me Maybe by Carly Rae Jepsen" [5] getting over 50 million views.

One thing that these viral youtubers have in common, is that they used another person to sing songs for them. And they did this manually by finding all the required pre-existing video and audio, and editing it manually to create the results. But what if this process was automated?

Machine Learning has come a long way, and is already capable of creating deepfake video [6][7] and audio [8] of specific speakers as they wish. The app Wombo.AI [9] has allowed users to upload an image of a face, which then used Machine Learning to animate the image as if it were lip syncing to a selection of songs they have prepared. These technologies could potentially be used to create covers of songs automatically with minimal effort, where it might look and sound like the person is actually singing.

However, for this thesis we focus on the process used in the aforementioned Baracksdubs YouTube channel. In Figure 1 it shows Barack Obama speaking part of the chorus of "Call Me Maybe by Carly Rae Jepsen", by concatenating clips of the lyrics when they are spoken during his public speeches. This method of cataloguing and concatenating clips to fit into a song is what will be automated in this thesis, by creating a pipeline that performs all the inter-

mediate steps. By automating this process, the creation of these videos will be significantly sped up. On top of this, as soon as one speaker has a large enough database of clip, it can be applied to many songs in almost no time, allowing for easily finding good combinations songs to fit with a speaker. The general objective of this thesis is to create a pipeline to complete this process quickly with good results. Once the pipeline is functional, it will be easier to create and compare videos with different input parameters to the effect of each of them on the result.

Figure 1: Barack Obama singing "Call Me Maybe by Carly Rae Jepsen", made by baracksdubs.

2 Related Work

2.1 Automatic Speech Recognition

Part of the pipeline will require the automated transcription of speech, also called Automatic Speech Recognition (ASR).

Tara N. Sainath et al. [10] from Google Inc. with their second-pass LAS decoder End-To-End system, score a Word Error Rate (WER) of 4.8% for Long utterances [11] (audio fragments longer than 5.5 seconds) on their dataset containing Google's voice search traffic in the United States, including added noise for robustness. The WER is a metric used in speech recognition that gives a value representing the amount of the transcription that is inaccurate. However the specific dataset it was trained on might not work well within the context of this thesis.

A potentially more generically applicable ASR near state-of-the-art is SpeechStew (W. Chan et al. 2021) [12], a Model trained on a broad variety of datasets, with WERs between 1.3% and 9%. SpeechStew is based on the idea that training a model on as much data as possible gives great results. Some fine-tuning or adaptations might be required for new tasks, which is relatively low cost.

2.2 Forced Alignment

Forced alignment is the act of aligning text and speech to get accurate boundaries for every word or phoneme.

R. Fromont and J. Hay's LaBB-Cat [13] is a browser-based open-source tool that allows for forced alignment. It is based on Hidden Markov Model Toolkit [14]. According to Gonzalez et al. [15] having an excellent score in both Overlap rate (The detected duration overlap of a word compared to a human benchmark) and Boundary Displacement (Comparison of the temporal boundaries between human and the forced aligner). For our purposes, their interface is less than ideal while the alternative, the Montreal Forced Aligner, is better suited.

The Montreal Forced Aligner (MFA) (2017) [16] utilises Kaldi [17] instead of the Hidden Markov Model Toolkit [14], allowing for it to be available as a stand-alone command line tool. MFA is based on the Prosodylab-Aligner, and performs better [16] than it and FAVE [18]. It features triphone acoustic models to capture the context in phone realization, and allows for re-training of the acoustic models on new data. It has a chance to mismatch the lyrics which quickly derails the entire alignment beyond salvageable.

2.3 Speech separation

Speech separation tries to separate the voices in audio files from the rest of the sounds, including noise and music.

Wavesplit [19] by N. Zeghidour and D. Grangier is an end to end speech separator that specifically aims at proper separation of likewise sources from one another, in this case voices from other voices. This is useful for songs with

multiple vocal tracks, and performs at the same level as the state-of-the-art for most common speech separation benchmarks in both a clear and a noisy setting.

For a faster and similarly succesful speech separator, C. Sabukan et al. [20] created SepFormer, a modern Transformer-based neural network which allows for high performance due to its parallelization capabilities. Also getting near state-of-the-art quality while significantly lowering memory cost and reducing training time.

Spleeter [21] is a more easily accessible and usable pre-trained command line separation tool supporting the splitting of audio into up to 5 different tracks (vocals, drums, bass, piano, and other). With close to state-of-the-art and up to a 100 times real time processing, this will separate songs in a couple of seconds with high accuracy. It has the advantage of being readily available and allowing for command line usage, which makes integration into the pipeline relatively simple.

2.4 The Speech-To-Song pipeline

Diana Deutsch et al. [22][23] found that speech and song are related quite closely, showing that repeated spoken phrases can start to sound more like singing. This could occur more easily here given that there will be actual matching music in the background, especially given the commonness of repeated phrases in songs. The melody created by the repeated phrases might not agree with the actual melody of the song, nor may the transitions between words be smooth enough to encourage the melodic illusion.

Yarn.co [24] has the ability to find quotes from movies, music, and TV. It allows for limited queries of specific people, and these will always be limited to finding the exact and complete phrases used in their movies. This is not a service to mix together clips to create new sentences, and only uses clips from movies and TV-shows. It has no regard for the timing of every word, and is presumably manually created. The videos returned by your query might include different people, because the database per individual is not large enough to cover a significantly large portion of the English vocabulary.

Crumbles.co [25] seemed similar at first sight, allowing any sentence to be made. However, after diving into the archives of the currently offline service, it was limited to only Homer Simpson clips, which are most likely handpicked. This does not allow for the scalability the pipeline of this thesis will.

Finally, Talk Obama To Me [26] is a service that lets Barack Obama say anything in the same way this pipeline does, and replaces missing words with phoneme reconstructions. No details have been posted about the creation of this website, but it seems to have defined one clip per word which suggests it was manually done, and therefore has the same scalability as Crumbles.co. If a website like this would be made for other speakers in the future, it could greatly benefit from part of this pipeline since it will create a database of clips for any video, allowing for quickly finding a great clip per word.

3 Design

As a proof of concept, a prototype of the Speech-To-Song pipeline is designed which will help us think about and understand the process step by step. The pipeline tries to automate a lot of the process from the speaker's video files and song's audio file, to the final product. The only manual steps in this design are acquiring the top-level files; Footage of the speaker, and the song's audio file.

Every green block indicates an intermediary result (usually a file) that is created and can be re-used to skip over the earlier steps. For example, if there is already a "Dataset with timings for every word of the speaker" you can then apply it to any song without having to transcribe footage again. If there already is a "TextGrid file with all lyrics and their timings", then that song can be sung by any speaker that has their own dataset of clips. This design is not limited to any specific implementation for any of the steps, making it easy to replace parts of the pipeline if another method with better performance is found.

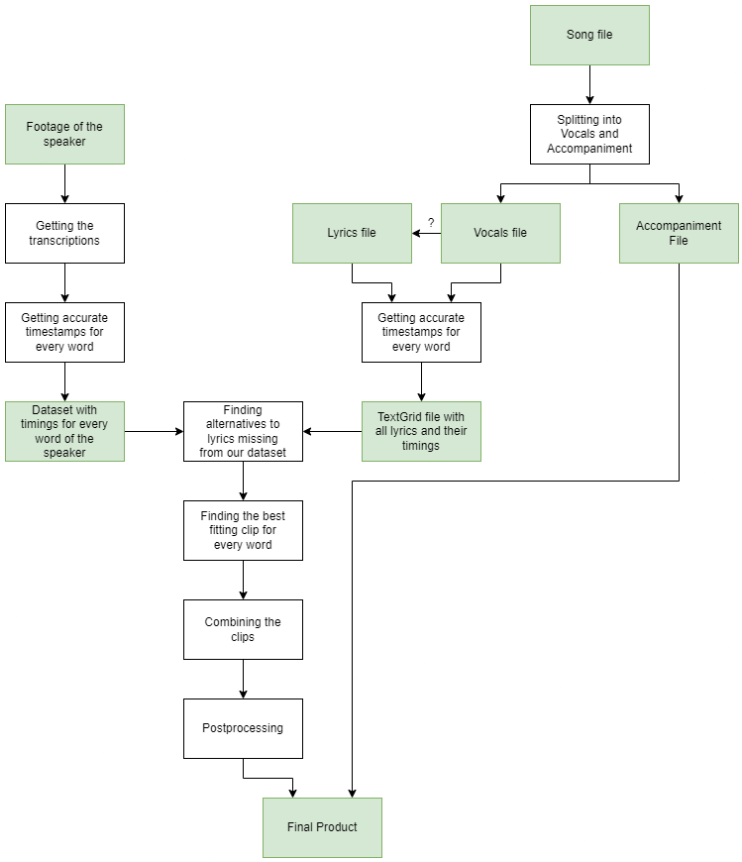


Figure 2: Pipeline design diagram.

The right side of the design contains all the steps pertaining to the song that will be performed by our speaker. This part of the process is incredibly fast in both the manual process and the automatic process that will replace it. The relatively small file size and duration mean that neither a human nor a program will have a hard time working with it.

The left side of the design looks shorter than the right side, but is relatively intensive work. Gathering the footage must be done manually, and since multiple hours of footage are most likely needed, this can require gathering dozens of videos. The process of getting the right timings of every word said by the speaker will be different for the manual and automatic versions of this part of the pipeline, so they will both be described.

If this footage will be manually transcribed, the process will likely compare the specific song lyrics to the text in the video, and only pick words that appear in the lyrics, as well as limit it to one clip per word. This means that the steps from "Getting the transcriptions" down to and including the "Finding of the best fitting clip for every word" will all be done simultaneously by the user, which is efficient. However, as the more common words are catalogued, the missing words will require increasingly more footage to appear. At this point the user can potentially search for videos that are more likely to contain the missing words to save time and use that to finish off the cataloguing process. For

The automated part will, once set up, automatically transcribe every word said in the video and create a large database of all the clips and their timings. This will take a lot longer than the song part of the pipeline, but will be significantly faster and easier than manually transcribing it. This will require that the footage is cleared of any other speakers, unless the transcription can properly distinguish between speakers. The result of this will contain all of the spoken words, rather than just the ones that are in the lyrics of the current song, meaning any future song will already have most of the work done. With a properly working method for replacing missing words, this setup process needs to be done just once per speaker. Looking at all the steps taken, collecting and cleaning the footage will be the most effort and time-consuming of the automated version of this left side of the diagram.

4 Research Questions

The objective of this thesis is to automate the aforementioned video creation process, to be able to look at how well it performs, as well as what the quality of the results are. Since a primary advantage of automation in general is to speed up a process, it is interesting to see how fast one can achieve desirable results using the created Speech-to-Song pipeline. And since the finding and cleaning of the speaker footage takes up the plurality of the time in this process, the main research question is:

What effect does the amount of input footage have on the quality of the resulting video made using the Speech-to-Song pipeline?

By looking for the answer to this research question we will be investigating the main bottleneck of the pipeline which can lead to optimizations. Our hypothesis is that an increase in input footage amount will increase the number of different words available which will result in a higher *word coverage* (meaning the percentage of lyrics in the song that have a corresponding clip available), and thus in a higher quality result. This effect most likely diminished over time. It should be noted that this hypothesis contains two assumptions, which directly correspond with the two sub-questions:

1. *How does the amount of input footage affect the word coverage in songs?*
2. *Does an increase in word coverage mean an improved quality of the result?*

5 Approach

5.1 Pipeline implementation

Figure 2 shows the steps in this process and how they connect to each other. This subsection will go into detail on the implementation of the steps the pipeline performs.

5.1.1 Footage of the speaker

Barack Obama will be the main speaker featured in this thesis because of the large amount of high-quality footage available, the fact that he speaks English, and to stay in-line with the original inspiration for this thesis.

To minimize the footage quality's effect on the results, the footage was selected to have high clarity audio of the speaker. On top of this, every video has been gone through manually to remove sections with high levels of background noise and audience interference. This is to ensure that the quality of our input footage is a constant.

5.1.2 Transcription

The ability to get good transcriptions from the videos is key to the quality of the results of the pipeline, and should be applicable to any footage. Ideally the transcription service could be done by an offline program included in the pipeline. For this purpose, PocketSphinx [27] and Mozilla's DeepSpeech 0.6.0 [28][29] are tested for accuracy and ease of integration. Mozilla's DeepSpeech required hardware I did not have access to at the time, luckily PocketSphinx seemed to function properly. When running the pre-trained PocketSphinx model on the vocals of the song however, the returning results were not even recognisable as the lyrics of the song. The results of speaking into the microphone were higher quality, so a potential reason for these results is the fact that they are sung, which might not work well with the way the model was trained. Either way the quality of the resulting transcription was significantly below what would be needed for this pipeline to function properly. Google Cloud Speech-To-Text [30] is an online service with the same purpose, and allows for enough free usage to meet the requirements of this thesis, and the resulting transcriptions are very high quality. Where PocketSphinx got nearly every word wrong, Google Cloud Speech-To-Text got only a few wrong.

On top of this, there is a YouTube Transcript API offered a method of gathering the subtitles of a specific video, which sometimes contains manually transcribed text which is ideal, and sometimes contains auto-generated subtitles that offer similar quality as the Google Cloud Speech-To-Text transcriptions with higher performance, since they don't have to be transcribed in the moment. However this method is possibly not publicly supported by YouTube, as mentioned on the API's website, which is not something that this pipeline wants to condone. It also means that the method can be taken down any day, which makes it unreliable, and on top of that it also assumes that the videos originate

from YouTube, which is not a dependency that is good for the general usability of the pipeline.

The Google Cloud Transcription service offered two different methods of getting transcriptions, long transcriptions (longer than roughly one minute) and short transcriptions. Using the short transcriptions had a disadvantage for the pipeline. The splitting of the files may interrupt the pronunciation of a word, meaning it may transcribe a word improperly. This can be counteracted by ignoring the first and last word of every segment, which means we will end up with a database smaller than it should be. However splitting the audio also has some big advantages. It allows for errors to not affect the results and the performance much, because one faulty file or transfer does not waste a lot of time nor does it affect most the rest of the videos. So when a connection error occurs when uploading the footage to the Google Cloud Service, only the relevant one minute segment need to be re-uploaded. If the speed of the transcription process is too slow, the split up footage can be sent in parallel to speed it up. On top of all this, the transcription seems to work roughly 30% faster when performed on small sentences, which may be related to the fact that Google has used short clips in their data set in the past to train their network. And finally, the splitting also removes errors in alignment step in the next section.

5.1.3 Accurate Timestamps

Despite Google Cloud's outstanding performance in transcribing speech, the timestamps it offers for every word are not accurate enough for the purposes of this pipeline. Another step would have to be done before a database of accurate clips can be created.

Speech alignment is required to get timestamps for every word that are accurate enough to be used by the pipeline, and the Kaldi-based Montreal Forced Aligner [15][16] provides exactly what is needed, a generically applicable command line based forced aligner that worked on Windows.

This provides us with .TextGrid files that contain timings for every word and every phoneme in that word. The combination of Google Cloud's Speech-To-Text and the Montreal Forced Aligner result in a reliable method of acquiring a dataset containing timestamps for every word using only the speaker's footage as input.

The alignment process may crash if it does manage to properly align the text with the audio, which has been hard to fix automatically. So the splitting of footage into segments allows for minimal impact when this does happen since it will only affect one of the segments without disturbing the others, allowing the pipeline to still function. On top of this it's possible for the aligner to get misaligned, and once it gets misaligned it quickly become a high severity mismatch, where one misalignment cascades into completely wrong results. Splitting up the tasks also provides a reset moment for this misalignment, allowing the cascading problem to be minimized.

5.1.4 Processing the song

Provided with only the song audio file as input, the pipeline uses the Spleeter's [21] pretrained source separation library to split the vocals from the instrumentation. This creates both an accompaniment file (i.e., a karaoke version) and a vocals file. The accompaniment file will be used later to provide the music in the final product.

The vocals file can both be used for the transcription of the lyrics, as well as the speech alignment to get accurate timings. However, the Google Cloud Speech-To-Text service produces significantly worse results using these files than it does with the footage of the speaker. These results seem to worsen significantly when background singers sing alongside the main voice, as well as when audio effects such as echo or distortions are used.

Due to the unreliability of this feature, the pipeline offers the user the option to either still get the automatic transcription, or to enter the lyrics themselves. This is shown in the diagram by the question mark above the arrow connecting the Vocals file to the Lyrics file, which will then be used by the Montreal Forced Aligner again to create a .TextGrid file containing the timings of every lyric.

5.1.5 Combining the speech and song

Now that the pipeline has both a .TextGrid file containing the lyrics of the song, and the dataset of clips spoken by Barack Obama, it can start to match these together. For every word it selects the clip that has a duration most similar to the duration in the song, while preferring longer clips over shorter clips because stretching out audio creates artifacts. But what if a word can not found in the main dataset at all?

Not every word sang in the song will have a corresponding clip spoken by our speaker. To resolve this, the user can look for more footage, but this process takes a lot of time and can still prove unfruitful if the available footage is limited or the missing words were never spoken by the speaker.

5.1.6 Missing words resolution

To resolve this problem, a more generically applicable solution needs to be explored. The importance of lyrics in song has been questioned a lot, where not everyone listens to the lyrics that are sung throughout most of the song. Lyrics also aren't always intelligible [31], which suggests the sound of the lyrics may be more important than the meaning of the lyrics themselves. Worth mentioning is that there's a large amount of people that do value the lyrics of songs [32], and the fact that this thesis is centered around the vocals might draw even more attention to the lyrics. Nevertheless, the suggestion that accurate lyrics may not be mandatory sparked the idea that replacing words with similar words could be a solution to the problem. As long as a word sounds similar enough, it may not be noticed by the viewer if a word is different from what it would normally be. Depending on the occurrence of the word, it may play a very insignificant and short role in the lyrics, which may fool even people that

do pay attention. There are also other ways to improve the odds of fooling viewers, as suggested by McGurk and MacDonald [33] in "Hearing Lips and Seeing Voices", speech recognition in humans is not an exclusively auditory experience. The expectation of a word by people that already know the lyrics, or the addition of subtitles to the song using the correct lyrics may convince the viewers that the word that was replaced is actually still there, so the final product will contain burned in subtitles.

To query for similar words, the API of Datamuse [34] is integrated into the pipeline. This allowed for several types of queries, including looking for *homophones* (words that are pronounced the same way), *rhyme words*, *similar sounding words*, as well as providing the number of syllables in each result. On top of these queries, recreating the word from phonemes is also an option, as well as searching for *superwords* (words that contain the original word as a subword).

Method	**Suggestion**
Homophone	Bare
Rhyme	Pear
Similar sounding	Beer
Superword	Bears

Figure 3: Examples of word replacement suggestions using the different replacement criteria on the word *bear*

Phoneme recreations were made using the phoneme data created by the Montreal Forced Aligner, in the same format as the timings of the words are. Using this timestamp data to combine the sounds of a words and recreate the original word can be used to replace any missing word. However, the allowable error in the timings is a lot smaller here than it is for regular words, and this turns out to be a big problem for this method. The resulting audio is usually not very similar to the original word, and the resulting video is very erratic. The results could become better by manually selection which of the phoneme timings are accurate, however this does not align with the purpose of this pipeline.

Superwords are a selection of words that contain the original word as a part of it. Since the pipeline selects words based on exactly matching the spelling of words, different forms of a word such as conjugations and plurals are ignored. As a partial solution to this problem, the idea of looking for words containing our missing word was created. The superwords should not be too different as that will most likely introduce more syllables and well as change the overall pronunciation. A big problem with this method is that the pronunciation might be completely changed with the addition of even a single letter (e.g. *bear* sounds different from *beard*).

The Datamuse API allows for more focus on pronunciation. To know which of the alternatives suggested by all these possibilities should be prioritised, the quality of the results they produce must be assessed. For some of these queries

it is clear how their quality compared to others; Homophones are a best-case scenario and should have the highest priority, and automated phoneme recreation give the worst results here. Words with the wrong amount of syllables are also generally too different, so all Datamuse queries are filtered to contain the same number of syllables as the word that needs to be replaced.

Due to the scope of this thesis, there is no extensive research done of defining the similarity suggested replacement words and the original missing word. The priority list that is created is based on a personal best estimate combined with the scores produced by Datamuse, as well as a small experiment with five participants comparing superwords to the best results of the Datamuse queries. In this experiment, replacement words generated using both methods were compared, and the participants were asked which replacement word was a better fit. From the experiment it became clear that superwords were generally inferior, with the exception of cases where the superword was the original word followed only by either *s* or *'s*, which occurred frequently.

Using this information, the pipeline selects the method of replacement in the following order:

1. Homophones
2. The missing word followed by *s* or *'s*
3. Words that sound similar and rhyme
4. Words that sound similar
5. Words that rhyme
6. Superwords with at most 3 extra letters.
7. Phoneme recreation

5.1.7 Finishing up

The resulting clips are inserted into an audio file at their respective timing in the song using FFmpeg [35], and the video clips are inserted into a video file using the same timings. The audio is then normalized for multiple passes using FFmpeg's normalization, as well as applying Spleeter voice isolation to combat normalization artifacts and artifacts created by the concatenation of clips. Subtitles of the original lyrics are added and as a final step, the audio file and video file are combined with the accompaniment file to create the final product.

5.2 Aspects of Evaluation

To be able to make a general statement about the effect of increasing the input footage on the quality of the video, the pipeline will be creating videos with different amounts of input footage, which will be called *versions*. The current pipeline has a dataset of 5600 unique words, and 64000 clips of all those words combined. To create *versions* with different amounts of footage available to them, *subsets* will be created of this main dataset, and the pipeline will then

create one *version* of a video per *subset*. Creating good subsets requires that the variables at play are considered, to potentially remove them from the equation or keep them in check so the experiment can be properly run.

5.2.1 Speaking rate

The *speaking rate* of the speaker is about how many words are spoken within a time frame. This includes differences in the speaker's words-per-minute count, as well as the amount of pauses the speaker takes during the footage (For example because the speaker is waiting for the audience to quiet down after an applause). If the the amount of input footage is to be defined solely in terms of time, these variables will greatly affect the words in our subsets.

To attempt to nullify the effect of this variable, this thesis be keeping track purely of the number of words spoken (e.g., 4000 words spoken), rather than the frame of time during which they were spoken (e.g., one hour of footage). This way the effect of pauses is completely negated, and the results are more easily relatable to other speakers, where they no longer must consider Obama's speaking rate when using the data of this experiment for theirs.

5.2.2 Vocabulary

Secondly the *vocabulary* is a noteworthy variable. This is a variable that itself is influenced significantly by multiple factors. One such factor is the speaker's own vocabulary, where speakers will phrase what they are saying differently based on their own knowledge and preferences. A second factor on this variable is the environment in which the footage is shot, because a person might speak less formally in front of friends than they would during their inauguration speech. And a third factor would be the topic about which they are speaking. A person could have a significantly above average vocabulary about a specific topic but have a below average vocabulary when speaking about something completely different.

On top of this vocabulary variable, it's also important how it matches with the vocabulary used in the song. Both the vocabulary of the songs writer as well as the topic of the song will have a big impact on the word coverage the speaker's footage will provide.

Since this variable has so many parts to it most of which are hard to quantify and isolate in an objective way, it is unrealistic to base the experiment around finding the effects of these variables on our results. Instead, there will be only one speaker (Barack Obama) in a relatively consistent setting for all the footage (Formally addressing large groups of people). There will, however, be different songs with different topics, moods, and singing speeds to get a broad estimate on if there's an effect on the quality of the result at all.

5.3 Generating subsets

Since the experiment will be using versions with varying amounts of input footage, it's important that the versions have subsets of footage that are a representative sample of what one might expect to find in a random piece of footage.

To this end, the process of how the words are selected from our main dataset of footage is key.

A straightforward approach to creating these subsets is mimicking the process as it would happen in practice. This would be through selecting consecutive sections of footage, as one would do when adding footage, and keep adding sections like this until the subset limit is reached. This will give a result that could happen when using the pipeline in practice, however it will skew the data towards the topic that the speaker happened to be talking about during this selected time span. Given the large amount of footage with unique topics, this will be a variable that can only be averaged out by creating many different variations for the experiment. Given the scope of this thesis it will be more useful to create a subset that is like an "average subset", so that the randomness of the selection process is reduced.

A method to create a more average set is the Monte Carlo method. In this method, word clips will be randomly picked from the main dataset until the subset limit is reached. This picking process will be automatically weighted by the larger number of clips available for common words, increasing the frequency at which common words are picked so that the subset will have a distribution statistically similar to the distribution of the main dataset. The most common words selected by this method are most likely to be words that will appear in any of our speaker's footage, which means that results created using this can also be expected to be similar to what one would create in practice. This is the method that will be used in this experiment.

To compensate for the random nature of this selection process there will still be multiple *variants* for any specific version of footage. But since this process will be less skewed than the mimicking reality method, it will require fewer versions to get the similar likelihood of conclusive results. This means the experiment will for each song have several *versions* with a different amount of input footage, and all these versions will have several *variants* to account for the randomness of the process.

5.4 Performance

The pipeline takes roughly five minutes to create the Call Me Maybe video of one minute with 137 words with a video resolution of 1280x720. The duration of the process scales with the amount of words and the duration of the video mostly, since most of the computational effort goes into the editing of the video. For creating a version more quickly, the pipeline allows for a lower resolution version, which roughly halves the duration of the process.

6 Methods

6.1 Experiment setup

The experiment will be using exponentially increasing step sizes (4000 / 8000 / 16000 / 32000 / 64000) because of a preliminary test, seeing what percentage of the top 1000 English spoken words in music you can expect to be spoken by the speaker, given certain amounts of input footage.

Random subsets are creating for a specific numbers of words out of the footage, and this is repeated 10,000 times for each. The results show that the relation between amount of input footage and these top 1000 words have positive yet clearly diminishing returns, where our exponential increases in the amount of input footage result in roughly linear increases in word coverage.

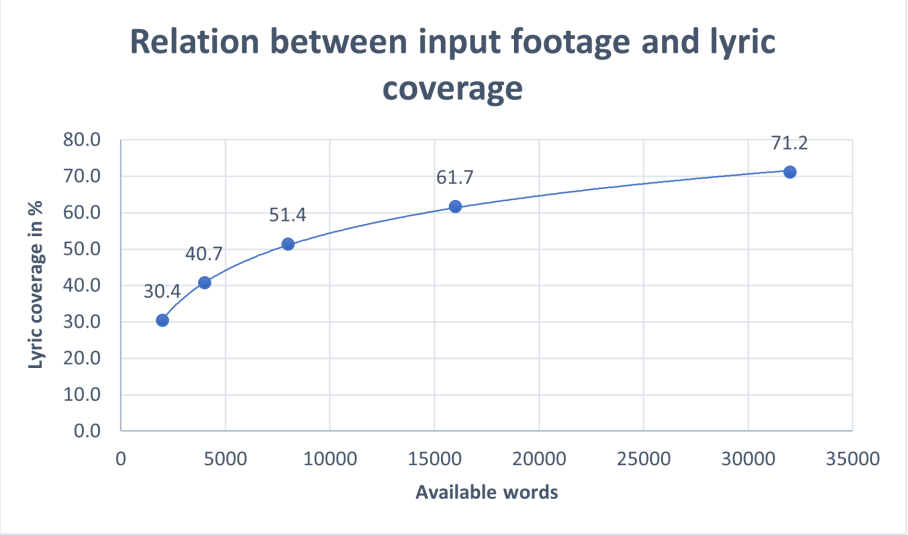


Figure 4: Relationship between input footage and lyric coverage compared to the top 1000 most common English words in music [36].

Therefore, it was concluded that going for linear increases in the amount of footage will result in either too small steps (resulting in requiring a vast number of different videos with minor differences between them), or too big steps (resulting in insufficient insight into the growth on the lower end of the scale).

Three songs are selected for the experiment; Call Me Maybe (CMM) by Carly Rae Jepsen, which is chosen because it was the original viral video of Barack Obama singing. Lose Yourself (LY) by Eminem, chosen because of its high number of words per minute as well as the similarity between rap and regular speech. And Imagine (I) by John Lennon, for its low word per minute singing and the serious mood of the song.

With three songs, five versions per song and five variants per version, a total of 75 separate videos are created. With the goal of creating a short experiment to increase the amount of people willing to participate, creating an experiment with one version for each song seems like a natural conclusion, because this creates an experiment that takes roughly five minutes total and has no songs repeat per person.

To minimize bias caused by the combination of videos in one experiment, the videos are split up into groups of three with a randomized version and variant per song. On top of this the order in which the songs are shown were also alternated to average out a potential bias caused by which video is seen first. All of this was with the constraint that each variant can only be in one triplet, and that every version must occur an equal amount in total.

The distribution of videos can be seen in Figure 5. The videos are described by a combination of the abbreviation of the song name and the amount of input footage used to create the video. The different shades are used to highlight the fact that the songs alternate the order of showing. Every occurrence of the same version is a different variant from the rest.

Group	First Video	Second Video	Third Video
1	CMM 8000	LY 32000	I 8000
2	LY 8000	I 64000	CMM 64000
3	I 64000	CMM 32000	LY 8000
4	CMM 32000	LY 64000	I 4000
5	LY 32000	I 8000	CMM 64000
6	I 16000	CMM 4000	LY 4000
7	CMM 64000	LY 64000	I 16000
8	LY 16000	I 64000	CMM 16000
9	I 4000	CMM 32000	LY 64000
10	CMM 64000	LY 64000	I 8000
11	LY 8000	I 16000	CMM 4000
12	I 64000	CMM 64000	LY 4000
13	CMM 8000	LY 4000	I 32000
14	LY 32000	I 4000	CMM 4000
15	I 16000	CMM 16000	LY 16000
16	CMM 4000	LY 16000	I 32000
17	LY 4000	I 32000	CMM 8000
18	I 4000	CMM 16000	LY 16000
19	CMM 32000	LY 8000	I 8000
20	LY 32000	I 32000	CMM 16000
21	I 4000	CMM 8000	LY 32000
22	CMM 32000	LY 16000	I 8000
23	LY 4000	I 16000	CMM 4000
24	I 32000	CMM 8000	LY 8000
25	CMM 16000	LY 64000	I 64000

Figure 5: The video groups.

6.2 Questionnaire

To answer the main research question, the questionnaire needs to ascertain the perceived quality of the videos. The original intent of these types of videos is to be entertaining, so the question in the questionnaire should reflect this. Part

of maintaining the entertainment factor, is to not need multiple viewings per video, and to not ask for properties that require attention to detail because those might detract from the entertaining experience that is the main goal.

To create a question that accurately estimates a user's experience, the questionnaire draw inspiration from the commonly used Net Promoter Scores' [37] main question, rephrased to better fit the informal context of viral entertainment videos. The latter half of the statement was added to not suggest users to actually share it, which could mess up the distribution as well as make them feel like they have to share the video if they agree with it.

"I would share this with a friend or family member if I saw this video outside of this research context"

The perceived entertainment of the video can also be reliant on whether the participant knows the original song, so they are asked to agree or disagree with the statement

"I know the original song"

Finally, to get an indication of how noticeable incorrect lyrics are, how well the word replacement feature disguises the bad lyrics, and how much the flow is retained even with replacement words the questionnaire asks participants how much they agreed or disagreed with the following statements:

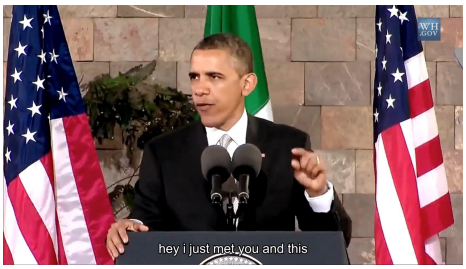
"I noticed a lot of missing or incorrect lyrics"

"The missing or incorrect lyrics affect the flow of the song a lot"

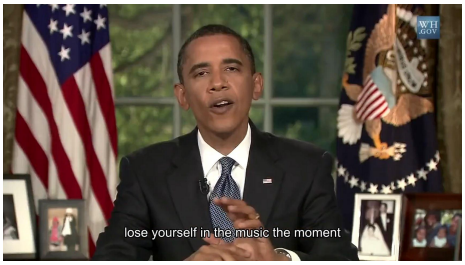
6.3 Distributing the experiment

To keep track of every response and the group they belong to, 25 forms are created corresponding to the 25 groups in Figure 3. The invitations for the experiment is to be sent through a mailing list, and to ensure ease of participation while distributing the forms equally, it would be ideal to only send one link and to have a system in place that would handle distribution automatically. To achieve this, a "master form" is created that contains the introduction of the experiment and a link to one of the forms. This URL is replaced by the least submitted form's URL through a spreadsheet that would receive all the responses, and updates whenever a new submission is sent in.

In Figure 6, three songs are displayed to provide an example of how the pipeline works. These videos were created with the full dictionary of clips containing roughly 64000 individual word occurrences. Links to the rest of the videos can be found in the appendix, including the amount of words used in the dictionary's subset in each video.



Call Me Maybe



Lose Yourself

Imagine

Figure 6: Video's of all three used songs created by the pipeline.
(Video's might not play properly depending on the software used to view this)

7 Results

7.1 Collected data

A total of 77 people participated in this experiment. Since the experiment did not ask for any personal details, the exact demographics of the participants is unknown. However, given that the experiment invitations were sent by email to master students at the University of Utrecht. The expected demographic of the participants include men and women, mostly age 21 and above, with an education level of at least a Bachelor student, that live in or near Utrecht.

Due to a delay in the updating of the spreadsheet that kept track of all the responses, some groups have more responses than others. In Figure 7 these numbers of responses are noted. Every form was completed by at least two participants, however the inconsistency surrounding group 25 is explained by the fact that the biggest mailing list was invited after already 23 responses had been submitted. This large number of simultaneous participants combined with the updating delay in the spreadsheet caused group 25 to have too many participants, as well as affecting group 24 and some of the lower groups.

The effects of this on the number of responses per version can be seen in Figure 8. Given the layout of the songs, every song still has the same number of participants, however number of responses per version vary between 13 and 20.

Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Responses	4	4	3	3	4	3	3	4	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	4	6

Figure 7: Number of responses per group.

Call Me Maybe		Lose Yourself		Imagine	
Version	Responses	Version	Responses	Version	Responses
4000	14	4000	14	4000	13
8000	16	8000	16	8000	15
16000	17	16000	14	16000	14
32000	13	32000	15	32000	15
64000	17	64000	18	64000	20

Figure 8: Number of responses per version.

7.2 Evaluation

7.2.1 How does the amount of input footage affect the word coverage in songs?

This sub-question was already tested in a general case in the Methods section, on the top 1000 most common English words in music. To confirm that this preliminary test also applies to this experiment, a Spearman's correlation test can be run on the amount of input footage and the word coverage in the generated videos (Figure 9). The Spearman's correlation test can show if there's

a correlation between two variables, and gives an indication of how strong this correlation is. The resulting correlation coefficient shows whether the results are positive or negative as well. In this case, a positive correlation coefficient would show that an increase in input footage likely correlates with an increase in word coverage. The Spearman's correlation test assesses relationships without requiring this relationship to be linear unlike the Peason correlation test. Since the amount of input footage increases exponentially rather than linearly, the Spearman's correlation test is a a better choice for finding the correlation here.

Figure 9 indeed shows that there's a positive correlation between these two. As well as giving us a p-value, which indicates the likelihood that these results happened by chance. Usually a p-value below 0.05 is accepted as a threshold where it is reasonably certain that the results were not by chance, so a p-value of less than 0.0001 suggests that it is very unlikely to be a mistake.

Spearman correlation test

Correlation	
Variables	*Coverage*
Input Footage	**0.521**

p-values	
Variables	*Coverage*
Input Footage	**<0.0001**

Figure 9: A slimmed down version of the Spearman's correlation matrix and the corresponding p-value matrix, between the amount of input footage and the word coverage on the generated videos.

7.2.2 Does an increase in word coverage mean an improved quality of the result?

To indicate the quality of the videos the participants were asking how likely it is they would recommend this to friends or family, which will be referred to as the Recommendation Score. Testing this second sub-question can be done by running a Pearson correlation test on the Word Coverage and the Recommendation Scores given by the test participants, the results of which can be seen in Figure 10. Like the Spearman's correlation, the Pearson correlation test shows indicates the correlation between two variables. Because both variables are on a linear scale, this allows us to use the Pearson correlation test instead. The results and the p-values can be interpreted in a similar way to the Spearman's coefficient.

The results of the Pearson correlation test show that there is likely a positive correlation between the Word Coverage and the Recommendation Score in Call Me Maybe, with a p-value of 0.020. The high p-values for the other two songs suggest that this correlation cannot be assumed for these songs. This implies that one of the other variables in these songs are affecting the Recommendation Scores more than the Word Coverage (and thus more than the amount of input footage).

Pearson correlation values			
	CMM	LY	I
Coverage	0.265	0.009	0.150

Pearson correlation p-values			
	CMM	LY	I
Coverage	0.020	0.940	0.192

Figure 10: The values and p-values of the Pearson correlation test, comparing Word Coverage with Recommendation Score.

7.3 Answering the main research question

What effect does the amount of input footage have on the quality of the resulting video made using the Speech-to-Song pipeline?

Videos were categorized by the number of words available in the subset used to create them, these categories were 4000 words, 8000 words, 16000 words, 32000 words, and 64000 words. To determine if there is a statistically significant difference between the means of these categories, a one-way ANOVA was used on each of the songs individually using XLStat [38]. The ANOVA results indicate whether there's a significant difference between the means of two or more groups. This is used to notice if there are actual differences in values based on the categories they are split up into.

As seen in figures 11, 12, and 13, the p value is not below the threshold alpha value of 0.05. This means in standard practise that based on these statistics it cannot be disproven that there is no correlation between the amount of input footage and the recommendation score of the results. The great difference in p-values between songs align with the differences found in sub-question two. The difference between the two could be explained by the fact that the amount of input footage also affects the amount of clips available, which in turn affects the selection process of clips that make it into the videos. This extra bit of variation might explain the difference between the results of sub-question two and the main research question.

Call Me Maybe

Source	DF	Sum of squares	Mean squares	F	Pr > F
Model	4	9.262	2.316	2.248	0.072
Error	72	74.166	1.030		
Corrected	76	83.429			

Figure 11: ANOVA Call Me Maybe's relation between the amount of input footage and the quality of the resulting video.

Lose Yourself

Source	DF	Sum of squares	Mean squares	F	Pr > F
Model	4	1.727	0.432	0.520	0.721
Error	72	59.728	0.830		
Corrected	76	61.455			

Figure 12: ANOVA Lose Yourself's relation between the amount of input footage and the quality of the resulting video.

Imagine

Source	DF	Sum of squares	Mean squares	F	Pr > F
Model	4	4.159	1.040	1.208	0.315
Error	72	61.971	0.861		
Corrected	76	66.130			

Figure 13: ANOVA Imagine's relation between the amount of input footage and the quality of the resulting video.

7.4 Discussion

There are multiple variables that could explain the inconsistencies between songs. Prior knowledge of the song seems unlikely to be the cause, since the multiple 2 sample t-tests comparing Recommendation Scores of participants with and without prior knowledge all have p-values suggesting significant uncertainty compared to the standard alpha of 0.05 (Figure 14). Worth noting is the low amount of participants without prior knowledge when comparing it to Figure 8, which could be the cause of these high p-values. Any versions not in Figure 14 did not have participants without prior knowledge of the song.

2 samples t-test on prior knowledge and Recommendation Score

Song version	LY 4000	LY 16000	LY 32000	LY 64000	I 4000	I 16000	I 32000	I 64000
Participants lacking prior knowledge	4	2	2	5	3	3	2	5
p-value	0.776	0.612	0.738	0.442	0.181	1.000	0.514	0.928

Figure 14: 2 sample t-tests comparing the Recommendation scores of participants with and without prior knowledge.

More likely, it's the result of one of the variables mentioned in section 4.2, which suggest that the vocabulary mismatch between the song and speaker, the singing speed, or the mood of the song could cause this. In particular, the lower score of Imagine could be explained by the more serious nature of the song resulting in a lower enjoyment factor and consequently in a lower recommendation score. And the lower scores of Lose Yourself could potentially be explained by the increased importance of lyrics on the flow of the song. However, these theories do not have any statistical basis.

8 Concluding remarks

8.1 Conclusion

The statistical analysis suggests that it cannot be confidently concluded that the amount of input footage influences the quality of the final product, meaning our main research question cannot be reasonable proven. There is no definitive correlation between amount of input footage and the recommendation score, however, there is a very likely correlation between the amount of input footage and the word coverage, as well as a positive correlation between word coverage and the recommendation score, answering both sub-questions. A possible explanation for these seemingly contradictory results could be that a recommendation score might not be the ideal way to approximate video quality in this context. This may have resulted in a not so accurate estimate of the quality of the final product, which combined with the apparent indirectness of their relation, caused the p-value to rise just above the acceptable threshold of uncertainty.

8.2 Future work

For more conclusive results, the experiment in this thesis could be adapted and redone with more specific criteria for the final video's quality. Many variables mentioned in this thesis could also make for interesting research. The effects of genre, mood, or singing speed on quality and enjoyment factor of the results could be interesting.

Using the pipeline for different languages should already be possible, but the support for other languages is limited by a mismatch in the language support of the services used in this pipeline. Creating a German version of the pipeline is possible, but the lack of Datamuse support as well as the incompleteness of the German pronunciation dictionary on the MFA website results in a video with no replacements for the missing words. The dictionary is used by the MFA to detect and potentially re-create words from phonemes, so when a word is missing it can't be properly detected. If this is improved then the differences between languages could be tested.

The word replacement method used by the pipeline contains a lot of different options, the relative quality of each could also be researched to improve the overall quality of the pipeline's output. (A perfectly functioning phoneme combination method could potentially replace the others). Combined with the visual part of DeepFake, it could look like actual footage and fit in with the rest of the video.

Creating a machine learning version of this pipeline could also be very interesting, and if trained to work with very little input footage this could perform very fast as well.

References

[1] J. Stafford covering "Jambalaya" by H. Williams www.cashboxmagazine.com/archives/50s_files/19521011.html

[2] Tim Ingham. (2021 Feb 24). Music Business Worldwide https://www.musicbusinessworldwide.com/over-60000-tracks-are-now-uploaded-to-spotify-daily-thats-nearly-one-per-second/

[3] M. Gregory, A. R. Gregory, E. Gregory, S. F. Gregory, Schmoyoho. (2010-present). Youtube. https://www.youtube.com/channel/UCNYrK4tc5i1-eL8TXesH2pg

[4] F. Saleh, Baracksdubs. (2012-present). Youtube. https://www.youtube.com/c/baracksdubsfans

[5] F. Saleh, Baracksdubs. (2012, June 4). Barack Obama Singing Call Me Maybe by Carly Rae Jepsen [Video]. YouTube. https://www.youtube.com/watch?v=hX1YVzdnpEc

[6] S. Suwajanakorn, S. M. Seitz, I. Kemelmacher-Shlizerman (2017, August) "Synthesizing Obama: learning lip sync from audio" https://doi.org/10.1145/3072959.3073640

[7] R. Anderson, B. Stenger, V. Wan, R. Cipolla. 2013b. Expressive visual text-to-speech using active appearance models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3382–3389.

[8] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu. (arXiv 2019) "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis" https://arxiv.org/abs/1806.04558

[9] B. Benkhin, P. Loungani, A. Jagga, A. Arneja, P. Pavel, V. Bhakta (2021). Wombo.ai. https://www.wombo.ai/

speech recognition. arXiv preprint arXiv:1507.06947.

[10] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T.r Strohman, Y. Wu, I. McGraw, C. Chiu (arXiv 2019 Aug 29) "Two-Pass End-to-End Speech Recognition" https://arxiv.org/pdf/1908.10992.pdf

[11] Malik, M., Malik, M.K., Mehmood, K. et al. Automatic speech recognition: a survey. Multimed Tools Appl 80, 9411–9457 (2021). https://doi.org/10.1007/s11042-020-10073-7

[12] W. Chan, D. S. Park, C. A. Lee, Y. Zhang, Q. V. Le, M. Norouzi. Google Research. arXiv:2104.02133 (2021) "SpeechStew: Simply Mix All Available Speech Recognition Data to Train One Large Neural Network"

[13] Fromont, Robert, Hay, Jen. (2012). LaBB-CAT: An annotation store. Proceedings of the Australasian Language Technology Workshop. https://www.researchgate.net/publication/282661289_LaBB-CAT_An_annotation_store

[14] Cambridge University Engineering Department. Hidden Markov Model Toolkit (HTK) https://htk.eng.cam.ac.uk/

[15] Gonzalez, Simon, Grama, James, Travis, Catherine. (2020). Comparing the performance of forced aligners used in sociophonetic research. Linguistics Vanguard. 5. 10.1515/lingvan-2019-0058.

[16] McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., & Sonderegger, M. (2017, August). Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In Interspeech (Vol. 2017, pp. 498-502).

[17] Povey et al. (2011) "The Kaldi Speech Recognition Toolkit" IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. Available on https://kaldi-asr.org/

[18] I. Rosenfelder, J. Fruehwald, K. Evanini, and J. Yuan, "FAVE(Forced Alignment and Vowel Extraction) Program Suite [Computer program]," 2011, available at http://fave.ling.upenn.edu

[19] Neil Zeghidour, David Grangier (2020, July) arXiv. "Wavesplit: End-to-End Speech Separation by Speaker Clustering". Found on https://doi.org/10.48550/arXiv.2002.08933

[20] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, J. Zhong (2020) "Attention is All You Need in Speech Separation" arXiv:2010.13154

[21] Hennequin, R., Khlif, A., Voituret, F., & Moussallam, M. (2020). Spleeter: a fast and efficient music source separation tool with pre-trained models. Journal of Open Source Software, 5(50), 2154.

[22] Deutsch, D., Henthorn, T., and Lapidis, R. Illusory transformation from speech to song. Journal of the Acoustical Society of America, 2011, 129, 2245-2252.

[23] Tierney, A., Dick, F., Deutsch, D., and Sereno, M. Speech versus song: Multiple pitch-sensitive areas revealed by a naturally occurring musical illusion. Cerebral Cortex, 2012.

[24] C. Butler, J. Krause. Yarn.co. Available on https://www.yarn.co/about

[25] Thirty Labs. Crumbles. https://www.youtube.com/watch?v=0xDNyOvMcSQ

[26] Ed King. Talk Obama To Me. (2016-present). Available on http://talkobamato.me/

[27] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar and A. I. Rudnicky, "Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices," 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, 2006, pp. I-I, doi: 10.1109/ICASSP.2006.1659988.

[28] R. Morais and developers. Mozilla. (2017-present) https://github.com/mozilla/DeepSpeech

[29] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng: Deep Speech: Scaling up end-to-end speech recognition. 2014 CoRR http://arxiv.org/abs/1412.5567

[30] Google LLC. Google Cloud Speech-to-Text. Available on https://cloud.google.com/speech-to-text/

[31] Nathaniel Condit-Schultz, David Huron; Catching the Lyrics: Intelligibility in Twelve Song Genres. Music Perception 1 June 2015; 32 (5): 470–483. doi: https://doi.org/10.1525/mp.2015.32.5.470

[32] Demetriou, A. M., Jansson, A., Kumar, A., & Bittner, R. M. (2018, September). Vocals in Music Matter: the Relevance of Vocals in the Minds of Listeners. In ISMIR (pp. 514-520).

[33] McGurk, H., MacDonald, J. Hearing lips and seeing voices. Nature 264, 746–748 (1976). https://doi.org/10.1038/264746a0

[34] Doug Beeferman. Datamuse. Datamuse API v1.1. (2016, December 5). Available on https://www.datamuse.com/api/

[35] Fabrice Bellard, FFmpeg developers. (2016). ffmpeg tool (Version 2761a7403b) [Software]. Available from http://ffmpeg.org/

[36] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011. Available on http://millionsongdataset.com/

[37] F.F. Reichheld, 2003. The one number you need to grow. Harvard Bus. Rev. 81, 46–54.

[38] Addinsoft (2022). XLSTAT statistical and data analysis solution. New York, USA. Available on https://www.xlstat.com/en.

9 Appendix

List of all the used videos. They are currently being hosted on YouTube which is not future-proof.

Song Name	#Words	Variant	URL
Call Me Maybe	4000	0	https://youtu.be/AVnVGE3uFp0
Call Me Maybe	4000	1	https://youtu.be/rXjnTI_E5FI
Call Me Maybe	4000	2	https://youtu.be/wGkys3SWLpg
Call Me Maybe	4000	3	https://youtu.be/g3d41_PLohU
Call Me Maybe	4000	4	https://youtu.be/J_4wo76P8bE
Call Me Maybe	8000	0	https://youtu.be/eAkLwqeYATQ
Call Me Maybe	8000	1	https://youtu.be/KdR4nUwCgps
Call Me Maybe	8000	2	https://youtu.be/K0b-zkiwbOg
Call Me Maybe	8000	3	https://youtu.be/9QfvgYlqTJ8
Call Me Maybe	8000	4	https://youtu.be/QoQD73At0EY
Call Me Maybe	16000	0	https://youtu.be/K4EIuANn-MI
Call Me Maybe	16000	1	https://youtu.be/jhu_n4ejIMY
Call Me Maybe	16000	2	https://youtu.be/uOnVCUtPmr8
Call Me Maybe	16000	3	https://youtu.be/LBaRerGb9lg
Call Me Maybe	16000	4	https://youtu.be/ht3HDUxVJYE
Call Me Maybe	32000	0	https://youtu.be/y-8vs0EEfuk
Call Me Maybe	32000	1	https://youtu.be/yYKD7RAyuZ0
Call Me Maybe	32000	2	https://youtu.be/Vf5awDSYYso
Call Me Maybe	32000	3	https://youtu.be/x9mvFpl_aKA
Call Me Maybe	32000	4	https://youtu.be/3QXrJPb2z2U
Call Me Maybe	64000	0	https://youtu.be/PXTFFT-ud0g

Song Name	#Words	Variant	URL
Lose Yourself	4000	0	https://youtu.be/8DHipmJ7I6k
Lose Yourself	4000	1	https://youtu.be/wlgZ37XO6w8
Lose Yourself	4000	2	https://youtu.be/KV5bJULMLF4
Lose Yourself	4000	3	https://youtu.be/gDq0ZmT-dAE
Lose Yourself	4000	4	https://youtu.be/HZ6AR6rHRUY
Lose Yourself	8000	0	https://youtu.be/iut59FS5D5Y
Lose Yourself	8000	1	https://youtu.be/IZgmdUdmzR0
Lose Yourself	8000	2	https://youtu.be/oHQ9ie73c-o
Lose Yourself	8000	3	https://youtu.be/ogMKWx6YC2c
Lose Yourself	8000	4	https://youtu.be/FvwwQ0IREOI
Lose Yourself	16000	0	https://www.youtube.com/watch?v=KmH27jMOtDI
Lose Yourself	16000	1	https://www.youtube.com/watch?v=2bIcQHyORgc
Lose Yourself	16000	2	https://www.youtube.com/watch?v=Eu3Mv3Q14Lo
Lose Yourself	16000	3	https://www.youtube.com/watch?v=H_3un610uO0
Lose Yourself	16000	4	https://www.youtube.com/watch?v=QHEpPMBNylc
Lose Yourself	32000	0	https://youtu.be/XhPD5f1rH80
Lose Yourself	32000	1	https://youtu.be/fRdO34Zgr74
Lose Yourself	32000	2	https://youtu.be/R2jhpCKVn-k
Lose Yourself	32000	3	https://youtu.be/79lYuUYL1gU
Lose Yourself	32000	4	https://youtu.be/PsZEinqxXKE
Lose Yourself	64000	0	https://www.youtube.com/watch?v=ad92_lQwQYA

Song Name	#Words	Variant	URL
Imagine	4000	0	https://youtu.be/K_P1MgK1V9o
Imagine	4000	1	https://youtu.be/L50pXBiRG-g
Imagine	4000	2	https://youtu.be/-9FpP9JePjE
Imagine	4000	3	https://youtu.be/QynhsqyZ87g
Imagine	4000	4	https://youtu.be/gyEGOQcscoY
Imagine	8000	1	https://youtu.be/9PE5V7ji6y0
Imagine	8000	2	https://youtu.be/Q45WHK1wQBQ
Imagine	8000	3	https://youtu.be/-TFBpKFhnZQ
Imagine	8000	4	https://youtu.be/Y3BoiL8PBhQ
Imagine	16000	0	https://youtu.be/B0yUkSe3qbo
Imagine	16000	1	https://youtu.be/326ATVNivM8
Imagine	16000	2	https://youtu.be/mYUODlBkrbI
Imagine	16000	3	https://youtu.be/KCxYgnSc3Wk
Imagine	16000	4	https://youtu.be/2nDnpcqg7lw
Imagine	32000	0	https://youtu.be/YX0JvddF8CE
Imagine	32000	1	https://youtu.be/r-gavX03RRw
Imagine	32000	2	https://youtu.be/0DXacGzinlw
Imagine	32000	3	https://youtu.be/CHwT9IEonsY
Imagine	32000	4	https://youtu.be/Yqdwqjpi09Y
Imagine	64000	0	https://youtu.be/H1DXylHtH20