Rappify: Adding Rhythm to Speech

Ian Torres
Stanford University
itorres@stanford.edu

Jacob Conrad Trinidad
Stanford University
j3nidad@stanford.edu

Abstract

This paper describes an automatic speech-to-rap synthesis system that can add rhythm to any given speech input. The system converts the speech input and instrumental beat input into a synthesized rap output. This is achieved by segmenting the speech input into individual words, and then rhythmically aligning these word segments. We implemented two methods of rhythmic alignment: aligning to a specified number of beats, and aligning to the acapella track of the original rap song. The performance of this system was evaluated using a subjective listening test with 8 subjects. The subjects rated the quality of several outputs of our system favorably, with the best output achieving a mean rating of 3.88 out of 5.

1 Introduction

Rapping consists of three main elements: content, flow, and delivery. Flow is made of rhythm, rhyme, and cadence. We present a system that, given any speech input and a piece of instrumental hip hop music (or "beat," for short), transforms the speech input such that it is rhythmically consistent with the beat. This allows the user to take any recorded speech, no matter the content and delivery, and transform it such it has adequate flow, as if the speaker was originally rapping rather than speaking.

To perform this task, we present a system that, for a given speech input and beat, splits the speech into word segments and then aligns these word segments with the beat using several techniques that we have discovered work well to simulate rhythm. We evaluate our system's outputs using subjective listening tests, with ratings on a 1 to 5 scale.

2 Literature Review

There is not a significant body of research on the task of speech-to-rap conversion. Speech-to-rap conversion is a special case of the general task of speech-to-music conversion, requiring a specific rhythm to be applied to the speech utterance that corresponds to the target instrumental track.

Saitou et al. propose a system for creating singing given a musical score, speech reading the lyrics of said score, and synchronization information associating each phone with a musical note (Saitou et al., 2007). Their system uses three main models: an F0 control model, a phoneme duration model, and a spectrum model. The F0 control model creates a target F0 contour from the musical score. The phoneme duration model uses the spectral envelope, aperiodicity index, and synchronization information of the speech input to lengthen or shorten each phoneme according to the length of its corresponding musical note. Lastly, the spectral control model adds singing to the speech by emphasizing the speaks found from the spectral envelope, dips from the aperiodicity index, and fits the changes to the target F0 contour. It is important to note that manual segmentation is needed to create the input synchronization information, preventing the speech-to-singing system from working automatically.

Wu et al. propose a system for converting speech to rap given the instrumental track and acapella track of a target rap song (Wu et al., 2014). Their system has two stages: analysis and synthesis. In the analysis stage, a beat tracking algorithm is used to obtain the positions of beats in the instrumental track. Additionally, a speech recognizer is used to obtain forced alignments of both the speech input and the acapella track with the lyrics of the rap song, at the word level. From the forced alignments, conversion factors for time-stretching and pitch-shifting the speech input to match the acapella track. In the synthesis stage, a phase vocoder is used to time-stretch and and pitch-shift the speech input. Finally, to create a "rhythmic effect", the volume of the modified speech input is increased according to the beat positions. This system achieved an a mean opinion score of 3.3 out of 5 on a subjective listening test.

There is currently only one publicly available application system that attempts to solve this task of converting speech to rap: AutoRap, a mobile app developed by Smule. Chordia et al. set to solve the task of automatically converting speech into song, rap, or other audible expression having target meter or rhythm. They approach this task by transforming spoken vocals "in accord with musical genres such as rap using automated segmentation and temporal alignment techniques".

The AutoRap system consists of 6 steps as shown in their patent (Chordia et al., 2013). The first step is onset detection, in which they compute a spectral difference function to find candidate onsets. The second step uses peak picking to discover strong onset candidates, allowing the system to segment the user input into chunks between two adjacent onsets. The third step is partition mapping. In this step, segments are partitioned into possible sub-phrases and mapped to possible phrase candidates using phrase templates. The fourth step is rhythmic alignment, in which a rhythmic skeleton is introduced for the given backing track to underlie the accent patterns. Using this skeleton, the phrase candidates are evaluated for rhythmic alignment and the best one is selected. The fifth step is timbre stamping, in which the selection is modified to better match the target song, such as being stretched, compressed, or pitch shifted. The last step is mixing, in which these selections are temporally aligned with the backtrack and the song is produced.

This system appears to be more complex then what is seen in Wu et al. with significant amounts of processing. This work does seem to pay off as the app AutoRap maintains an average 4.0 out of 5 rating on both iTunes and the Google Play Store.

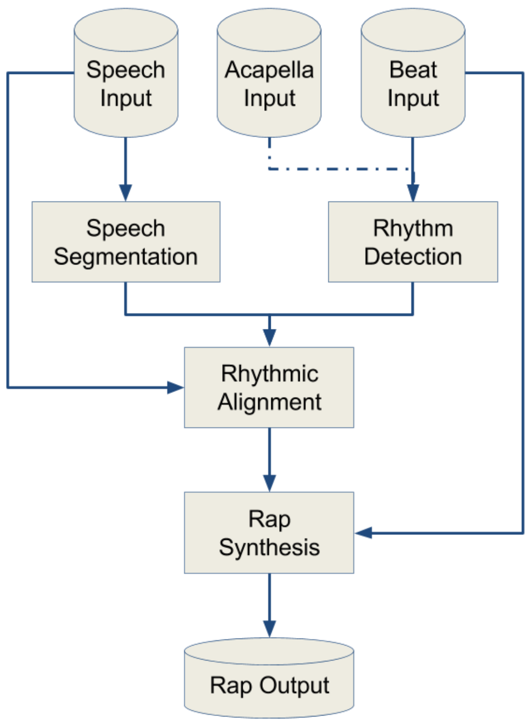

Figure 1: Base Rap Synthesis System. It takes a speech input, beat input, and an optional acapella input and outputs a synthesized rap.

3 Approach

The overall structure of our system design is shown in Figure 1. As inputs, our system takes audio files of the recorded speech, the desired beat, and optionally the acapella track corresponding with the desired beat. It then produces as an output an audio wav file, which is the synthesized rap of the given audio file on the beat.

The first stage of the system is speech segmentation. The goal of this stage is to split the given speech input into segments that will be used in the rhythmic alignment stage of the system. We implemented two methods of segmentation, which are described in section 3.1.

In the rhythm detection and rhythmic alignment stage, we first locate elements of rhythm in the beat input or acapella input. Then, we modify the speech input to fit the desired rhythm by using the time bounds of each word from the segmentation stage. We implemented two algorithms for this stage, which are described in section 3.2. We use a phase vocoder to speed up or slow down the speech input without pitch changes. The vocoder used is part of Sound eXchange (SoX), a command line utility for sound processing (Bagwell, 2015).

Optionally, we also repeat every seventh phrase in order to make the end of each 8-count more defined. This step enhances the sense of rhythm in the output, as it is a common technique in rap to repeat a word at the end of a measure.

In the final stage, we overlay the modified speech input and the beat input. The speech input begins playing after a specified number of beats to account for the beat input's "intro." The resulting audio file is our final output.

3.1 Speech Segmentation

Our initial method of segmentation splits the speech input by words. To achieve this, we use a speech recognition system and a forced alignment algorithm. First, we use the speech recognizer to obtain a transcription of the speech input. For speech recognition, we use PocketSphinx, a version of CMU Sphinx (Huggins-daines et al., 2006).

Given the transcript from the speech recognizer, we then run a forced alignment algorithm, taking the transcription from the speech recognition stage and the speech input audio as inputs. This forced alignment gives us the time stamps of the beginning and end of each word in the transcript, allowing us to segment the user's speech into words. To perform forced alignment, we use Aeneas to synchronize the audio with the text transcription (ReadBeyond, 2017).

The second method of segmentation splits the speech input by beats. We use a beat tracking algorithm to determine the locations of beats in the speech input. Note that while there no literal beats in the speech input, we use a beat tracking algorithm on the speech input in an attempt to detect the rhythm and onsets in the speech, thus segmenting the speech based on its natural cadence. The locations of beats in the speech input will then be where the speech input is split. The implementation uses a beat tracking algorithm from madmom, a Python audio and music signal processing library (Böck et al., 2016). The madmom implementation was the top performing beat tracker in the MIREX 2016 Beat Tracking task (eXchange, 2016). It works by determining the tempo of the input and then iteratively aligning the beat around its estimated position.

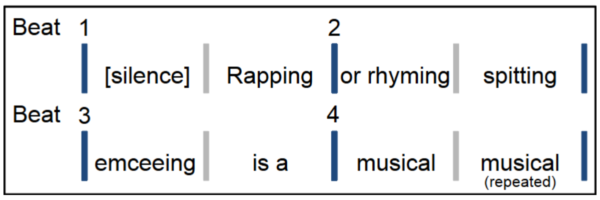

Figure 2: Four-beat snippet of the rhythmic alignment of the Wiki input from section 4.1, aligning to beats.

3.2 Rhythm Detection and Rhythmic Alignment

The goal of these two stages is to speed up or slow down segments of the speech input to give the speech input a sense of rhythm. We implemented two algorithms to accomplish this task.

Our initial algorithm aligns the speech input to beats in the beat input. In the rhythm detection component of this algorithm, we run the beat detection algorithm described in Section 3.1 on the beat input to find the time stamps of beats. In the rhythm alignment component, we align the segments of the speech input to the beat time stamps by speeding up phrases of words so that they fit exactly within a specified number of beats. In practice, we have found that fitting each phrase to either a single beat or a half-beat produces the best output. We do this by iteratively concatenating words into a phrase and stopping once the phrase's duration is longer than the desired number of beats. Figure 2 shows the result of aligning a given speech input using this algorithm.

The second algorithm aligns the speech input to the word alignments of the acapella track of the original rap song. In the rhythm detection component of this algorithm, we run the word segmentation method using using speech recognition and forced alignment on the acapella input as described in Section 3.1. In the rhythm alignment component, we speed up phrases of words to fit in each segment of the acapella track in the same way we did in the first algorithm. Note that this algorithm requires the additional input of an acapella track that corresponds with the beat track.

3.3 Evaluation

For our evaluation, we conducted a subjective listening test on a 30-second long sample output with 8 subjects. Subjects listened to the sample output. a 30-second clip of the original rap song, and a clip of the original speech utterance simply overlaid on

Speech Input	Beat Input	Repeat On	Rating Mean	Rating Std
Wiki	Humble	Yes	3.63	0.518
Obama	Forgot About Dre	Yes	3.50	0.535
Frost	Forgot About Dre	No	3.13	0.641
Seuss	The Next Episode	No	3.88	0.641
Stanford	Humble	No	2.5	0.535

Table 1: Initial Experiment Ratings ($n = 8$).

Speech Input	Base System	Beat Segmentation	Acapella Alignment
Good Day	2.38	3.00	3.25
Ocean	2.13	2.75	3.25
Cities	2.50	3.00	2.63
Email	3.50	2.88	3.00

Table 2: Beat Segmentation and Acapella Experiment Ratings ($n = 8$). Results shown are the rating mean for each speech input and system implementation.

the instrumental track. Subjects were asked to rate the sample output on a scale of 1 to 5, where 5 is the quality of the original rap song, and 1 is the quality of the original speech utterance overlaid on the instrumental. We record the mean and standard deviation of the ratings for each output.

3.4 Additional Segmentation Methods

We attempted to use other methods of segmentation. The first method was phrase segmentation using onset detection. The second method was phrase segmentation using silence detection. We had initial difficulty obtaining good phrase segments using either method and decided to stick with the methods described in Section 3.1. Future work could be done fully exploring these methods in an attempt to improve speech-to-rap systems.

4 Experiments[1]

4.1 Initial Experiment

Our base system uses the initial methods of word-based segmentation and beat alignment. We produced five raps for subjects to listen to and rate to test this base method. The five speech inputs we used were a reading of the Wikipedia article on rapping ("Wiki"), a clip of an interview with President Obama on Syria ("Obama"), a reading of of Robert Frost's Poem "Stopping by Woods on a Snowy Evening" ("Frost"), a reading of Dr. Seuss's book *The Cat in the Hat* ("Seuss"), and a reading from the "About" page from Stanford University's website ("Stanford"). The beats used were "Humble" by Kendrick Lamar, "Forgot About Dre" by Dr. Dre, and "The Next Episode" by Dr. Dre. The results of the subjective listening test can be seen in Figure 1.

"Wiki" was one of our speech inputs that was segmented and aligned very well, achieving a mean rating of 3.63. The repeat effect worked especially well for this input, as most of the repeated (half-)beats happened to contain individual words (e.g. "musical" and "backbeat"). The repetition of "musical" is particularly interesting because the three sylablles of "musical" are pronounced in an even triplet rhythm, and this complex rhythm is further emphasized by the repeat effect. There is also a decent amount of variation in the speed of the speech in the output; for example, "performed or chanted" is a lot slower than the following phrase "in a variety of ways." While too much fluctuation in speed would be jarring, in this case it is subtle enough to make the output more rhythmically interesting without disrupting the flow of speech.

"Obama" is an example of truly natural speech, as it is a prerecorded interview of President Obama speaking extemporaneously, whereas in the other inputs for this experiment, the speakers read from a script and knew that their speech would be used as inputs to our system. Thus, in comparison to the other inputs, Obama speaks a lot more quickly with less pauses in between words. As a result, many of the phrases that are fit to each beat consist of multiple words. In some cases, this makes

[1]Audio samples for these experiments and more are in the supplementary materials found in the submission and on the authors' websites.

the output more rhythmically interesting, but in other cases, the phrase is sped up so much that it is very difficult to understand. Nonetheless, “Obama” still had a decent mean rating of 3.50. It is possible that listeners simply think that faster speech sounds more like rapping. Another possibility is that the speaker being a famous figure with a very recognizable voice boosted the ratings as well.

“Frost” was deliberately read slowly and clearly in order to get as accurate a transcription and forced alignment as possible. Despite this, it scored a mean rating of 3.13, which was lower than the ratings of “Wiki,” “Obama,” and “Frost.” As the words are pronounced slowly with more silence in between them with natural speech, most of them are segmented properly and placed into their own individual beats. The resulting one-word-per-beat rhythm, with little variation in speed, might not be as interesting or aesthetically pleasing as some of the other outputs, which could explain the lower ratings in the listening test.

“Seuss” is a reading of a poem written in rhyming couplets with a consistent meter, and thus is already somewhat close to a rap. Thus, it is natural that this input achieved the highest mean rating of 3.88. However, the speaker did not read the poem to a strict beat, and the system clearly enhanced the speaker’s rhythm at several points. For example, the rhyming lines “it was too wet to play” and “all that cold, cold, wet day” both finish exactly at the end of a measure. Additionally, each word of “cold, cold, wet day” is placed in an individual half-beat, which accentuates the last word of the phrase. But even when the system does not align lines of the poem properly with the measures of the beat input, the rhyming words still naturally stand out to the listener, which “tricks” the listener into ignoring instances of poor rhythmic alignment.

“Stanford” is an example of a poor output of the system. Similar to the “Obama” speech input, this input contains few pauses between phrases in the speech. This caused a similar rap output, such that it is spoken fast and is then difficult to understand. The speech was also not articulated as clearly as “Obama,” hence the low evaluation rating.

4.2 Beat Segmentation and Acapella Alignment Experiments

We used four speech inputs for subjects to listen and rate in order to test potential improvements on our initial method by using either beat segmentation (see section 3.1) or acapella alignment (see section 3.2). The four speech inputs we used were a rap of Ice Cube’s “It was a Good Day” (“Good Day”), a reading of the lyrics of Ween’s “Ocean Man” (“Ocean”), a reading of the introduction of Charles Dickens’ “A Tale of Two Cities” (“Cities”), and a reading of an insurance email (“Email”). For all of the inputs, the beat input was the instrumental of Ice Cube’s “It was a Good Day” and the acapella input was Ice Cube’s corresponding acapella track. No repetition was used in this experiment. The results of the subjective listening test can be seen in Figure 2.

Given these results, “Good Day” and “Ocean” achieved their best evaluation when using the acapella alignment method, “Cities” achieved its best evaluation with the beat segmentation method, and “Email” achieved its best evaluation with the original base system. These trends can be explained when examining the original rhythmic nature of the speech input.

Both “Good Day” and “Ocean” are based on songs that contain rhythm and rhyme. Acapella alignment improved these speech inputs because they were able to be stretched to mirror the acapella track of Good Day because they were already musical. Note that the “Good Day” speech input didn’t align to its original nature in the rap because of our system’s preference to speed up the speech input, thereby causing it to be faster than the original rap. It is also likely that due to the elements of rhyme already embodied in the speech, the outputs sound more like rap because of these inline rhymes.

“Cities” is based on a rhythmic piece of text, with several poetic elements inherent in it. As a result, it did well with beat segmentation because the algorithm was able to detect this natural rhythm. When aligned with the beats, each line of the original text fit well within two beats, adding the appearance of rhythm.

“Email” has no inherent rhythmic elements in it. Because of this, beat segmentation fails to improve it. Furthermore, acapella alignment also does not improve the rap because the natural pauses in the acapella track aren’t natural in the speech input.

4.3 Forced Alignment Error Analysis

Since forced alignment plays an important role in the system, we developed two metrics to quantify its performance with respect to the rhythmic alignment. First, we define the split rate S of a given forced alignment to be the percentage of words in the speech input that were split due to incorrect time bounds.

$$S = \frac{\text{\# of split words in speech input}}{\text{total \# of words in speech input}}$$

Second, we define the combination rate C of a given forced alignment to be the percentage of non-silent segments in the forced alignment that contain two or more words from the original speech input.

$$C = \frac{\text{\# of segments containing multiple words}}{\text{total \# of non-silent segments}}$$

There are two noticeable qualitative errors in our system that can be attributed to inaccuracy in the transcription and forced alignment. First, certain words are not in rhythm with the beat input. This error can be correlated with the split rate, because if a word is split up in the alignment process, it will be placed across two beats at different speeds.

Second, certain phrases are sped up so much that they are incomprehensible. This error can be correlated with the combination rate, as if many words are aligned as a single word, they will likely have to be sped up with large factor.

Table 3 shows the split and combination rates for each input from the initial experiment. The Obama input's high combination rate explains the many phrases that are sped up too much. The most noticeable examples in the output include the phrase "in the opposition," which is sped up by a factor of 3.17 and the first occurrence of the phrase "chemical weapons," which is sped up by a factor of 2.52. Played in isolation, both of these modified segments are very difficult to understand.

Likewise, the Wiki input's high split rate explains why many words do not fall directly on the beats. For example, in line "rhyme, rhythmic speech, and street vernacular," the words "speech" and "vernacular" are both split among three separate segments. This results in this line having a rhythmically awkward delivery in the output. In comparison, the words "rapping," "rhyming," and

Speech Input	Split Rate	Combination Rate
Wiki	21.6%	25.0%
Obama	13.6%	51.9%
Frost	7.3%	7.7%
Seuss	4.1%	13.5%
Stanford	18.9%	18.8%

Table 3: Split and combination rates for each input from the initial experiment.

"spitting" at the beginning of the track all are segmented correctly, and so the individual words are each fit into individual half-beats.

However, a low split and combination rate does not necessarily correlate with the perceived quality of the rap. "Frost" achieved a 7.3% split rate and a 7.7% combination rate, but scored a lower mean rating than some of the other inputs.. This is possibly because Frost was deliberately read very slowly and articulately, and thus the resulting rhythm of the output was less interesting, as discussed in section 4.1.

4.4 Noisy Backgrounds

An additional issue with the current state of our system is a lack in robustness in handling noisy input. The system is unable to work well in noisy environments. As we currently do not attempt to remove background noise or manipulate volume in the speech input, the background noise remains present throughout the entire pipeline.

This issue can also be attributed to the fact that the forced alignment stage of the pipeline produces time bounds for words that sometimes contains silence before or after the given word. Given inputs with noisy backgrounds, these silences are replaced with background noise. This noise will then be heard in the synthesized rap output.

Furthermore, since the rhythmic alignment stage generally speeds up words and phrases to fit within beats, it will also speed up this background noise. When speeding up the speech with a noisy background, it makes the synthesized rap even more difficult to understand.

5 Conclusion

Our system can modify a given speech input to have a stronger sense of rhythm and cadence when overlaid on a beat. Our listening test results showed a favorable evaluation of the quality of our system's outputs for several inputs. Additionally, comparison of different methods in section 4.2 shows that the beat alignment method works better for natural speech, and the acapella alignment method works better for speech that already contains rhythm and/or rhyme. Our system does not work well for all inputs; this lack of robustness can be attributed mainly to errors in transcription and forced alignment, as well as our naive method of constructing phrases from the segmented words.

Future work can be done to improve split and combination error rates in the forced alignments by improving word segmentation. This can be achieved by adding noise reduction to enable its usage in louder environments and by improving the speech recognizer and forced alignment algorithm to obtain more accurate splits.

Other methods of segmentation can also be explored. It is possible to further explore lower-level segmentation by segmenting at the syllable level, which is probably more appropriate for this particular task. Musically, rapping is about arranging syllables over the beat in an aesthetically pleasing way, so we do not care as much about how the speech is separated at the word level. For example, syllable-level segmentation could potentially allow for more stressed syllables to hit the beats in the instrumental track.

Conversely, we could also explore higher-level segmentation by segmenting the speech at the phrase level. This approach would enable us to rearrange the speech input more naturally than at the syllable or word level and thereby potentially introduce choruses/verses into the rap. This process can be approached by splitting speech at silences or by using onset detection to find the start of stresses in the speech input.

Additionally, we could also take rhyming into consideration when segmenting the speech input. Assuming we could obtain a very accurate transcription, one method of incorporating rhyme would be to convert the transcription into phones using CMUdict, and then scan for words that have matching phones in their final syllable. (For example, by matching all phones from the last vowel onward.) Then, phrases ending in rhyming words could be extracted, rhythmically aligned to measures, and then rearranged to form rhyming couplets.

References

Chris Bagwell. 2015. Sound exchange (sox). http://sox.sourceforge.net/.

Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. 2016. Madmom: A new python audio and music signal processing library. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, New York, NY, USA, MM '16, pages 1174–1178. https://doi.org/10.1145/2964284.2973795.

P. Chordia, M. Godfrey, A. Rae, P. Gupta, and P.R. Cook. 2013. Automatic conversion of speech into song, rap, or other audible expression having target meter or rhythm. US Patent App. 13/910,949. https://www.google.com/patents/US20130339035.

Music Information Retrieval Evaluation eXchange. 2016. Mirex 2016: Audio beat tracking. http://nema.lis.illinois.edu/nema_out/mirex2016/results/abt/smc/.

David Huggins-daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alex I. Rudnicky. 2006. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *in Proceedings of ICASSP*.

ReadBeyond. 2017. Aeneas. https://www.readbeyond.it/aeneas.

T. Saitou, M. Goto, M. Unoki, and M. Akagi. 2007. Speech-to-singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices. In *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. pages 215–218. https://doi.org/10.1109/ASPAA.2007.4393001.

M. J. Wu, C. H. Lu, and J. S. R. Jang. 2014. Automatic conversion from speech to rap music. In *2014 International Conference on Electrical Engineering and Computer Science (ICEECS)*. pages 245–250. https://doi.org/10.1109/ICEECS.2014.7045255.