Automatic Generation of Hip-Hop and Rap Lyrics

A backwards n-grams model using its own corpus to produce original rap lyrics

Madeleine HERNANDEZ

Master 2 pluriTAL | 2017-2018

Dissertation (mémoire) supervised by Kim GERDES (Sorbonne Nouvelle – Paris 3)

Jury : Sylvain KAHANE, Serge FLEURY, Kim GERDES

Table of contents

List of Tables

List of Figures

Introduction

I. Goals

This study aims to answer the following questions:

1. Can a machine creatively produce hip-hop and rap lyrics?
2. Can the machine produce organic original lines?
3. Will the lines rhyme? How can the lines rhyme?
4. Will the lyrics be equivalent to modern hip-hop and rap lyrics?
5. Will the produced lyrics be semantically coherent?
6. Can the machine's output be recognized as human output?

These questions will be answered using several different methods with some trial and error to find the best solution. The results of this study were founded through several objectives. The overall objective of this study was to create a deep learning machine to formulate hip-hop and rap lyrics at the most naturalist outcome. Steps in the process of creating such a machine included: tackling the input line and producing an output verse—equivalent in rhyme, stress, and semantic meaning. A subgoal of this study was analysing the mechanisms in hip-hop and rap lyrics, dissecting the intricacies of the genre, such as, what constitutes a rhyme and the layering of rhymes involved in these lyrics. In addition, the importance of stress in combination with the phonetic transcription in a line was also involved. The goals are as follows:

1. Find related words connected to the query input.
2. Phoneticization of the found related words.
3. Search for rhyming words to use along with the related words.
4. Correlate themes in rhyming words through filtering.
5. Production of a verse including end-word rhymes with semantic coherence.

Before the study, this paper will begin with an introduction to the genres of hip-hop and rap, as well as a discussion about the new frontier of the genres. The following section is a brief history of rap and hip-hop. The section covers the founders and game-changing artists in this cultural movement, technological advances, and how this music affects other genres. After, there is a following section about text generation, which covers a study in the automatic generation of poetry—since poetry and rap share rhyming properties.

II. A Brief History of Hip-Hop and Rap

Hip-hop began its climb into the American mainstream in the 1970's. Rap, a part of hip-hop, had existed in some form long before the 1970's with origins tracing back to Africa, but it began increasing in popularity as an artistic expression during the rising hip-hop movement. According to Nelson George, the three grandfathers of hip-hop music are Kool DJ Herc (Herc), Afrika Bambaataa (Bambaataa), and Grandmaster Flash (Flash); they carved and defined the culture across an array of crowds through the records they played. Although the first examples of rap in the US came to life in the 1920's, Anthony Holloway, also known as DJ Hollywood, is recognized as the first disco rapper, a caterer to the mainstream *Disco Fever* of the 1970s. However, the first rappers of the hip-hop genre on the scene were those at Herc's parties, they chanted street phrases to the crowd and kept the party uplifted; these people are known as Masters of Ceremonies, or simply MCs, according to Grandmaster Flash's account (Forman & Neal 2012). The founders' recognize Melle Mel, as the first rhyming rapper, when he joined the Furious Five. They state, "Melle Mel and his brother Kid Creole were the first to really *flow* and have a poetic feel to their rhymes" often rhyming back-and-forth between each other (Forman & Neal 2012). This acknowledgement from the founders established the bar for rapping.

Despite creating the movement, none were responsible for the first ever hip-hop record. The first hip-hop record reached the mainstream in 1979, entitled *Rapper's Delight* by The Sugar Hill Gang. However, the originality of the lyrics is questionable because Big Bank [Hank] of The Sugar Hill Gang was the doorman at Sparkle (a now-closed bar and nightclub in New York in the late 1970s) where Grandmaster Casanova Fly, also known as Grandmaster Caz, performed; Big Bank never gave credit to Grandmaster Casanova Fly for the rhymes, according to Bambaataa (Forman & Neal 2012). Grandmaster Caz was another prominent rapper at the time who made his mark while playing with the Cold Crush Brothers, a rival group to The Furious Five, which included Melle Mel. Herc himself approached him [Hank] at his pizza shop job in New Jersey and told him, "When Sylvia Robinson, (the founder and CEO of *Sugar Hill Records*) hears the real deal, she gonna know" alluding to the fact that the real hip-hop rappers of the time were those in the Furious Five and the Cold Crush Brothers (Forman & Neal 2012), both being prominent groups who had also battled one another in *rap battles. Rap battles* are feuds performed among a crowd where opposing groups try to out rap the other and is often incentivized with a cash prize and bragging rights. Moreover, *Rapper's Delight* record's release impacted the genre; it began the commercialization of hip-hop and became more widely available to the public. The genre was being produced for the mainstream, no longer just for the crowds at clubs and parties.

Hip-hop, as a genre, stands out among others because of its complexity and uniqueness in form. This complexity was due in part to an array of artists in the late 70's and 80's. For example, the use of a *social narrative,* or *narrative sequencing*, was used by these MC's: Melle Mel, Slick Rick, Big Daddy Kane, and countless others. Narrative sequencing, including both ritualized storytelling and social narrative speech, is a frequently occurring genre in Black American discourse (Smitherman 1977). More lyrical complexities include, *flow* that is attributed to Rakim. Connor explains *flow* as the rhythmic structure that arises in

a rap from the interaction between the rapper's words and the strictly musical rhythms of those words.[1] These are just a few features that transformed the genre as it arose in popularity. Furthermore, rap's intricacies are rooted in how the lyrics are delivered over a *beat*, serving as a vehicle to deliver an artist's flow. The 1980's, or *The Golden Age*, was a transformative time giving birth to a handful of pioneers ranging across different aspects of hip-hop.

Hip-hop's beginning first saw MCs (Rappers, i.e. Masters of Ceremonies), as merely company to the real stars, disc jockeys, or DJ's. In the past, and to this day, MCs aid in curating parties; MCs include the audience through a *call and response* and this creates a musically-charged experience. *Call and response* is the manifestation of the cultural dynamic which finds audience and listener or leader and background to be a unified whole; as Black American culture stresses commonality and group experientiality, the audience's linguistic and paralinguistic responses are necessary to co-sign the power of the speaker's rap or call (qtd. in Forman & Neal 2012 538-9). The unison, better the connection, between an audience and the MC sparked a new genre–hip-hop. It is important to note that hip-hop is a cultural movement and rap is the genre, however, both are used interchangeably in this study. Its early start began with parties hosted by DJ's; the notorious DJ Kool Herc, one of hip-hop's founders, is recognized for playing enigmatic dance records at his parties in the Bronx, New York, the birthplace of hip-hop (qtd. in Forman & Neal 2012). It was at these community oriented parties where hip-hop began. Houston Baker quotes Jazzy Jay, wherein she describes hip-hop's musicality, "We'd find these beats, these heavy percussive beats, that would drive the hip-hop people on the dancefloor to breakdance. A lot of times it would be a two-second spot, a drum beat, a drum break, and we'd mix that back and forth, extend it, make it twenty minutes long" (qtd. in Forman & Neal 2012). This idea traces back to Herc because he used the *break*, or the drum beat, when playing and played one break after another break. He created an ethereal atmosphere removed from gang and street violence. This inspired other DJ's, and thus, catapulted the advent of rap in the broader hip-hop movement. MCs and break dancers heavily influenced these notorious parties and took on an important role in the movement, later becoming pillars of hip-hop.

The musicality provided by DJ's sparked a development of turntable technology and techniques. Grandmaster Flash, profoundly shaped DJ-ing techniques and his contributions are still used today; they are important because he was the first to implement them. He would later be associated with the rap group The Furious Five. The concept of scratching−the backbone of hip-hop DJing−came out of his laboratory (with an assist from his friend Grand Wizard Theodore); Flash's introduction of the "beat box" turned DJs from beat mixers to beat makers (qtd. in Forman & Neal 2012). His use of intertwining the scratching, the back-beat, and the punch phrasing created a unique sound from an already existent piece of music, in essence he created new music from old music, a combination never heard before. Before Flash began manipulating records, using his hands, this practice of touching the vinyl record was seen as taboo. In doing so he broke a cardinal rule, which transformed DJ-ing techniques for the subsequent generation of disc jockeys. This general notion of nonconformity is commonplace in rap and hip-hop music of the past and present.

[1] https://genius.com/posts/1669-The-rapper-s-flow-encyclopedia

Over time, MCs, also known as rappers, took on a greater role in hip-hop and started rapping to the beat which the DJ would spin. Rap, unlike spoken word poetry, is distinguished by its use of a *flow*, which relies on a beat, something not heavily used in poetry. Flow is how the MC navigates his or her words around the beat. Hip-hop appealed to the mainstream in the 1980's because of the transformative usage of lyrics and the first record, *Rapper's Delight*. The usage transcended in how beats were intertwined with storytelling in the *Golden Age* of rap. Shortly into Flash's career he joined a group, becoming known as Grandmaster Flash and The Furious Five. Their most significant song entitled *The Message*, released in 1982, was the first significant political recording produced in the post-soul era, representing an astute critique of the rise and impact of the Reagan right on working-class and urban locales (qtd. in Forman & Neal 2012 486). This social commentary, in *The Message*, was monumental because it eclipsed simplistic "house party" rhymes and inspired other artists to evoke more story-filled rhymes. DJ-ing and MC-ing are not the only pillars of hip-hop; breakdancing, graffiti, and fashion also began playing a more significant role.

As rap was constantly changing sounds from its origins of call and response to more simple lyrical rhymes, arose the *new sound of hip-hop*. The group Run-DMC is said to be the pioneer of this new hip-hop sound during the Golden Age (qtd. in Forman & Neal 2012 585). This group was part of the record label Def Jam Records, which was founded by Rick Rubin and soon after joined by Russell Simmons. This label heavily influenced the hip-hop culture, later becoming a pinnacle of recording labels. They were responsible for alternative hip-hop and rap artists like LL Cool J, The Beastie Boys, Slick Rick, and Public Enemy; all of whom are classified under the new school of hip-hop (qtd. in Forman & Neal 2004 541-550). These artists carved out their own style of hip-hop, which embedded rock undertones and a grungier sound. This suggested hip-hop as the new rock and roll. For example, the song *Walk This Way* originally by Aerosmith had been redone with Run-DMC and this collaboration brought hip-hop further into the mainstream inspiring new sounds and styles for the next generation of MCs. Hip-hop even started infiltrating the fashion world as Run-DMC's collaboration with Adidas was the first between a non-sports group and an athletic company. This is important because it was the start of hip-hop's direct influence on fashion and its commercialization.

Hip-hop as a sound and culture expanded in the late 1980's and 1990's when it reached Los Angeles. Ice-T, a Los Angeles rapper, who was heavily influenced by Schoolly D's gangster rap style, began writing Krip rhymes and eventually wrote *6 in the Mornin'* a street anthem and realistic rap song, which was Ice-T's first gangster rap (qtd. in Forman & Neal 2004 230). Hard-core gangster rap is an indication of a general crisis of morality in the black community; hip-hop, particularly gangsta rap, also attracts listeners for whom the "ghetto" is a place of adventure, unbridled violence, erotic fantasy, and/or an imaginary alternative to suburban boredom (qtd. in Forman & Neal 2012 111, 146). Lyrically just as abrasive, however, gangster rap turns the narrative of storytelling on its head by unpacking the gangster through the lens of state-sponsored racism. This complexity is credited to Schoolly D, originally from Philadelphia (qtd. in Forman & Neal 2004 150-1). This hip-hop style is heavily associated with Los Angeles, and the surrounding areas. The evolution of

this style of rap unfolded the birth of N.W.A. (*Niggaz Wit Attitudes*) a very famous hip-hop group known for their gangster raps. The political climate of the late 80s and early 90s in Los Angeles greatly influenced this group. N.W.A was composed of Arabian Prince, Dr. Dre, DJ Yella, Eazy-E, and Ice Cube. As a group, they impacted American culture because of their realistic gangster raps, and open opposition to the police and police brutality towards minorities; this sentiment was realised in a song entitled *Fuck The Police* (qtd. in Forman & Neal 2012 185). This impact even prevented the performance of this song at their concerts because of the risk of altercations with police.

Around the same time in the early 90's Dr.Dre would produce the album *The Chronic*. A solo-project album and it would be considered the first time hip-hop welcomed melodies and R&B. This album also alluded to a production breakthrough compared to other albums because it did not rely on sampling (qtd. in Forman & Neal 2004 208-10). This album has been noted to have brought hip-hop completely into the mainstream and subsequently the mainstream accepted hip-hop for what it was (qtd. in Forman & Neal 2004 165-6). This album and countless other hip-hop and rap albums have shaped the mainstream culture to this day, along with artists, record labels, producers, and more. This brief history of hip-hop and rap has laid the building blocks and inspiration envisioned for this project. It is useful for a few of the following sections concerning the creation and transformation of the corpus and analysing rhymes and techniques present in rap lyrics.

III. Machine generated poetry

Machine-generated poetry has its own history that includes several different studies that lean towards machine-generated rap lyrics. The following study described below sets a precedent for uses in machine and deep learning to produce poetry and can be related to this current study on machine-generated rap lyrics. The example set forth illustrates how poetic output can be undetectably produced by machine learning algorithms.

The State of the Art

The study entitled *Generating Topical Poetry* uses a combination of tasks to automate poetry generation. These tasks include the use of vocabulary, related words, rhyme words, finite state acceptors (FSAs), and path extractions (Ghazvininejad et al., 2016). They also extended their poetry generator beyond English by including the Spanish language. The poetry in the study was generated through the user input of words by way of creating a system that generates poems with topical coherence and rhyme schemes, and this extends across different poetry formats (Ghazvinineja et al., 2016). In the study, their model bases the poetic output directly on the word or phrase implemented by the user. The study uses an FSAs with deep learning to achieve more formal, correct poems directly correlated to the input.

The first task of vocabulary was based on stress found in a word or phrase. Stress was symbolized with a 1 and no-stress with a 0. This task was given rules, wherein, ambiguous words were removed because the stress in these ambiguous words depended on the word's usage. The second task of topically related words and phrases uses three methods based on the user input. The three methods are: a scoring of one-thousand words

and phrases related to the input, a pointwise mutual information system, and a word-to-vector model trained on a continuous bag-of-words (BOW) with different window sizes (Ghazvininejad et al., 2016). The BOW model had been trained on text from Wikipedia with a larger window, which is most beneficial and exudes better results for this study. The sole purpose of this task was finding the most optimal words and phrases–1000 to be exact. They were to be scored based on correlation to the user-input and intended to be analysed for their rhyming capabilities in a poem in the following task.

The task of choosing rhyme words was split into four sections; the first two pertain to rhyme types, strict (masculine and feminine) and slant, and the latter two discuss non-topical rhyming words, and rhyme word selection. The first two sections base stress and pronunciation on the CMU pronunciation dictionary. They ultimately mitigate rules on possible rhymes for words. A strict masculine rhyme would share the stress positionally and would be present in the latter part of the word, for instance, 'p**it**' and 'm**itt**' whereas, a feminine rhyme shares the penultimate stress, such as, '**unc**ertain' and '**cur**tain' (Ghazvinineja et al., 2016). Strict rhyme classes were implemented for the masculine and feminine rhymes, while a rule-based system was created for slant rhymes because they are considered half rhymes. The last section of this task creates classes of rhymes for words and phrases; the optimal rhyme is saved for the beginning and end of a poem, with the ending taking precedence.

Once the rhymes have been chosen, a finite-state acceptor is created. The FSA compactly encodes all word sequences that use these rhyme words and also obey formal sonnet constraints (Ghazvinineja et al., 2016). The constraints of a sonnet are imposed using different states of the FSA and the different states record the line number and syllable count. These paths in each FSA state are then ranked with deep learning models.

This study uses a recurrent neural network language model, instead of an n-gram model since it often produces poor poetry (Ghazvininejad et al., 2016). The model they chose is better suited for automatically generated poetry because it yields less robotic outputs. They trained their language model on 94,882 English songs. Since they used songs they were forced to implement a penalty for repetition as well as a reward when generating less 'safe' words for the end of a line (Ghazvininejad et al., 2016).

The study also implemented another model, the translation model. This translation model emphasized the topic word. This was done through training an encoder-decoder sequence-to-sequence model. The translation model first chose the line ending words based on the topic (encoder side) and was then paired with the entire reversed lyrics (decoder side) (Ghazvininejad et al., 2016). This ensured the knowledge of the 14 rhyme words before generation.

The results of both models (mentioned above) were tested by an average of 24 human judges. The judges preferred the translation model to the language model by more than 20% among stanzas and sonnets. The latter difficulties faced in this study include an argument on plagiarism and the correct use of 'a' and 'an'. The former issue on plagiarism was argued to have been prevented through the use of the strong conditions on rhyme,

meter, repetition, and ambiguously stressed words, whereas, the latter issue on grammar was fixed in a post-processing step, as errors were prevalent (Ghazvininejad et al., 2016).

The output from their poetry generator, in a *grand* sum, takes a query word, or phrase, and produces a poem. Here is a piece produced with their algorithms:

Query:	*tangerine*
Couplet:	A smoke was dripping from an orange *jelly*,
	Started screaming at the singing **sands**.
	I had a belly full of bloody *Mary*,
	Naked on the side of Indie **bands**.

The poem presents and ABAB rhyme scheme with lines that are semantically coherent to the theme of the citrus fruit–a tangerine.

Rhyme Types and Schemes in Rap and Poetry

This section explores the different forms of rhyming in poetry and rap. Section I describes the requirements for each rhyme form and provides an example. Section II uses a sonnet, a classical form of poetry, to elucidate some of the descriptions provided in Section I and provides some contrast to the rap excerpts presented in Section III. Section III demonstrates several different sets of rap lyrics taken directly from the corpus along with their phonetic counterpart (see *Corpus Construction*, Section III for *Phonetic Transformation*). It also dives into the types of rhymes present in rap lyrics and how they overlap with one another.

I. Rhymes Types

This is a quick exploration of the different rhymes types present in both poetry and rap, including examples; it is meant to be used as a reference for the subsequent sections.

A. Strict and Slant Rhymes

Strict rhymes diverge into feminine and masculine rhymes. In a masculine rhyme, the last syllable is stressed; in a feminine rhyme, the penultimate syllable is stressed (Ghazvininejad et al., 2016). The penultimate syllable is the second to last syllable in a word and a feminine rhyme often has two syllables.

- Masculine: In this phrase the words '*hate*' and '*a*' share the same final stress and an assonance rhyme.
  - *You see me comin', get your ass runnin', I know you **hate** **a** huey*

- Feminine: The words '*comin*' and '*runnin*' exemplify a feminine rhyme through their presentation of two syllables and primary stress on the first syllable.
  - *You see me **comin'**, get your ass **runnin'**, I know you hate a huey*

A slant rhyme is also called a half-rhyme, a near-rhyme, an imperfect rhyme, and an oblique rhyme. This rhyme type is very common and occurs where syllable end sounds are similar but not identical (Brown and Hirjee, 2009).

- Slant: A half-rhyme is present with the end words '*lips*' and '*lit*' as they don't end with the same sounds but are similar enough to rhyme.
  - *Yo as the blunt hangs low from my defiant **lips***
    *Smoke emerge in the shape of halo's the chronic's **lit***

B. Internal, Multisyllabic, and Perfect Rhymes

An internal rhyme occurs in the middle of a line and it can span multiple lines. In a perfect rhyme, the words share exactly the same end sound (Brown and Hirjee, 2009). A normal (perfect) rhyme involves two syllables that share the same nucleus (vowel) and coda (ending consonants) (Malmi et. al, 2016).

- Internal: The internal rhyming of '*didn't*' and '*kickin'* is within the line's boundaries.
  - *I **didn't** have any role models **kickin** the truth out*

- Perfect: Both '*back*' and '*track*' share the nucleus and ending consonants.
  - *But I'm **back** on **track**, add to the fact that I never really drive sober*

Multisyllabic rhymes are rhymes spanning over several syllables across one or more words. Rap music often features triplets or longer rhymes with unstressed syllables following the initial stressed pair, which may span multiple words (Brown and Hirjee, 2009). In fact, multisyllabic rhymes are hallmarks of all the dopest flows, and all the best rappers use them (Malmi et. al, 2016).

- Multisyllabic: The rhyming of '*pumpin*' and '*fear*' allude to a multisyllabic rhyme.
  - *I'll make you're life easier, cuz I'll be **pumpin fear***

C. Bridge, Chain, and Compound Rhymes

Bridge rhymes are internal rhymes spanning across two lines (Brown and Hirjee, 2009). Chain rhymes, or holorimes, are consecutive words or phrases in which each rhymes with the previous word or phrase (Brown and Hirjee, 2009). Compound rhymes are formed when two pairs of line internal rhymes overlap within a single line (Brown and Hirjee, 2009).

- Bridge: The internal rhyme in the first line is being extended into the second line.
  - *Red **diamonds** he **designin'** got me **shinin'***
    *Spotlight cover **my** body, **my** chain **blindin'***

- Chain: Among these lines, is the evidence of a chain rhyme as the rhyme '*Rover*' is linked back to '*sober*' in the previous line. The same could be said for '*racks*', '*back*', and '*track*'.
  - *Now it's racks on racks, uh, never thought that I would ride **Rover***
    *But I'm back on track, add to the fact that I never really drive **sober***

- Compound: The intersection of '*shit*' and '*in*', an internal rhyme, with another internal rhyme '*weighs*' and '*a*' creates the perfect environment for a compound rhyme.
  - *The shit **weighs** out in **a** month, the loss goes on timeless*

D. Assonance and Consonance Rhymes

Assonance is the repetition of vowel sounds and can span multiple words (Kao and Jurafsky, 2016). Consonance, the contrast of assonance, is the repetition of consonant sounds in a word and it can also occur across an array of words (Kao and Jurafsky, 2016).

- Assonance: The same vowel is shared in '*walls*' and '*balls*'.
  - *Flat screens on my **walls**, flos look like bowling **balls***

- Consonance: The consonant sound ‘T’ is present throughout the phrase and consecutively in ‘*stick it out till*’.
  - ***T**ook an oath I'm a **st**ick i**t** ou**t** **t**ill the end*

E. Alliteration

Alliteration is the repetition of a sound, or letter, at the start of a word across more than one word; it is a rhyming tool present in both poetry and rap lyrics.

- Alliteration: The words ‘*be*’ and “*bitin’*” both share the same starting letter ‘B’.
  - *Why you always gotta **be** **b**itin' my lip*

II. Poetic Rhymes

Poetry is a minimal form of art; it relies on vocal or written expression. However, poetry is quite complex when broken down; it includes several different genres each emitting different emotions, rules, and fervour. The same could be said about rap as it has expanded genres, going beyond hip-hop. To name a few, a piece of poetry can be: a ballad, a haiku, a sonnet, a blank canvas, a limerick, a free verse, an elegy, an epic, an ode, etc.

Sonnets are interesting because of their notoriety and the rules involved in this genre. Sonnets are most famous because of their relation to the actor, poet, and playwright William Shakespeare. In fact, there is a slight contrast between the general notion of Italian sonnet and a Shakespearean sonnet. This is important because it alludes to how he personalised the poetic art form, affecting the genre, which is of interest because rap has similarly evolved from the contributions of several notorious rappers, such as Rakim. This form of poetry, just like others, is notable because of the dictating rules in the poetry’s meter.

A sonnet is composed of fourteen lines partitioned across three four line stanzas and an ending couplet. A couplet is a succession of two lines, most often at the end of a poem, whose final words rhyme and often share the same length, or meter. The couplet plays a significant role in this form of poetry because of the rhyme scheme, which is equally important. Meter is the number of stressed and unstressed syllables in a line; iambic pentameter is the specific meter used in a sonnet. Iambic pentameter uses five stressed feet and five unstressed feet, one after the other, for a total of ten syllables all together. Moreover, a rhyme scheme is the pattern presented by each line’s final word in a song or poem. There are many different rhymes patterns present today, but the one often associated with a sonnet is the pattern “ABAB CDCD EFEF GG”; it is also the same pattern used in Shakespeare’s works. Other patterns exist depending on the type of poetry or even the poet, for example a limerick will have this pattern “AABBA”, whereas a ballad will have three stanzas with this pattern “ABABBCBC” and the final stanza ending with this pattern “BCBC”.

(1) Be wise as thou art cruel; do not *press* (A)
My tongue-tied patience with too much *disdain*; (B)
Lest sorrow lend me words and words *express* (A)
The manner of my **pity**-want**ing** *pain*. (B)

(2) If I might teach thee **wit**, better **it** *were*, (C)
Though not to love, yet, love, to tell me *so*; (D)
As testy sick men, when their **deaths** be *near*, (C)
No news but **health** from their physicians *know*; (D)

(3) For if I should despair, I should grow ***mad***, (E)
And in my madness might speak ill of *thee*: (F)
Now this ill-wresting world is grown so ***bad***, (E)
Mad slan**derers** by mad **ears** believed *be*, (F)

(4) That I may not be so, nor thou *belied*, (G)
Bear *thine eyes* **straight**, though thy proud **heart** go *wide*. (G)

FIGURE 1: Shakespeare's *sonnet 140*.

In the sonnet, there are examples of alliteration, assonance, bridge, internal, multisyllabic, slant, and strict rhymes. Alliteration is demonstrated throughout the sonnet in these smaller phrases: '*tongue-tied*', '*teach thee*', '*my madness might*', '*wresting world*', and '*though thy*'. In the third stanza, assonance rhyming is present in the first and third lines because 'mad' and 'bad' share the same 'a' vowel sound. An example of a bridge rhyme includes these two lines, "*The manner of my **pity**-want**ing** pain*." followed by "*If I might teach thee **wit**, better **it** were*", where 'pity', 'wit', 'it', and the last syllable of 'wanting' rhyme internally across more than one line. The last line in the third stanza emanates an internal rhyme where the last syllable of 'slander**ers**' slant rhymes with '**ears**'; in the last line of the sonnet there is another slant rhymes involving 'str**aight**' and 'h**eart**'. The third line in the first stanza exhibits a multisyllabic rhyme, '*Lest sorrow lend me words*', where '**Lest** sor**row**' rhymes with '**lend** me **words**'. The parts emboldened represent the rhyming. As for strict rhymes, there is evidence only of masculine rhymes and none of feminine rhymes, such as, the one demonstrated in '**thi**ne' and '**e**yes' where the first and only syllables rhyme with one another–making them a masculine rhyme.

These are just a few of the rhyme types mentioned in the previous section (Section I). The others not mentioned and even those discussed in this section will now be presented with relation to rap lyrics in the next section (Section III).

III. Hip-Hop and Rap Rhymes

Rap shares certain qualities with poetry, however, they differ in terms of measurement. Rap uses bars, whereas poetry uses meters. Poetry relies on the quantity of feet in the line's meter, but rap employs sixteen bars per verse—a standard. To quote Rakim, the notorious MC, "I try to start with sixteen dots on a paper and if four bars were this long I see like a graph in between them four bars; I could place so many words and so many syllables. I could take it to a point where there's no other words you can put in that four bars."[2] In a rap song with a four-by-four time signature, there is at least one verse; a verse,

[2] https://www.vox.com/2016/5/19/11701976/rapping-deconstructed-best-rhymers-of-all-time

or colloquially a hook, is composed of eight or sixteen bars and one bar is a grouping of four beats. A verse can be filled with either a set up bar or a punchline bar with the setup most often preceding the punchline bar; they are both useful types of bars holding a multitude of different types of rhymes. A bar can be seen as an equal and another term for a line (of rap or poetry) and is often composed of ten to sixteen syllables on average.

Here is an example of a perfect rhyme from Nate Dogg's song *These Days* featuring Daz with a CMU phonetic gloss underneath (see *Corpus Construction*, Section III for *Phonetic Transformation*).

---

*Cause everybody knows my **name***
K AA1 Z, EH1 V R IY0 B AA2 D IY0, N OW1 Z, M AY1, **N EY1 M**

*All the police think I'm servin **game***
AO1 L, DH AH0, P AH0 L IY1 S, TH IH1 NG K, AY1 M, S ER1 V IH0 N, **G EY1 M**

Across these lines, '*name*' and '*game*' share the same coda, 'EY', and end sound, 'M', thus creating a rhyme link among these lines and forming a perfect chain rhyme. A perfect rhyme is the most basic and commonly used rhyme in rap since its genesis; it is the same as an end rhyme. Moreover, if these two lines possessed their own rhyme scheme at the end of a verse they could be seen as a couplet.

Here is an excerpt of Timbaland's *Lobster & Scrimp* featuring Jay-Z and an abundance of assonance rhymes highlighted in different colors for readability.

---

*Now these **bitches wanna give me more** head **than** Sade*
N AW1, DH IY1 Z, **B IH1 CH IH0 Z**, **W AA1 N AH0**, **G IH1 V**, **M IY1**, **M AO1 R**, HH EH1 D, **DH AE1 N,** , SH EY1 D AH0

*I need **privacy** I pull **up the back** shade*
AY1, N IY1 D, **P R AY1 V AH0 S IY0**, AY1, P UH1 L, **AH1 P**, **DH AH1**, **B AE1 K**, SH EY1 D

Assonance rhyming is illustrated three times in these lines. The first instance is the 'IH' sound across '*bitches*' and '*give*' where both fall within the line's boundary, making this an internal assonance rhyme. The following is with the 'AH' sound present in '*wanna*', '*privacy*', '*up*', and '*the*'. Additionally, this 'AH' assonance rhyme's extension into the second line makes it an assonance bridge rhyme. The last assonance rhyme is also a bridge rhyme, spanning the two lines, with the 'AE' sound in '*than*' and '*back*'. Furthermore, alliteration is evident in the first line with the letter 'M' in the words '*me*' and '*more*', which follow each other sequentially. These are just some rhyme types present in rap. Despite being simple in form they are fundamental for smoother rhymes.

The album Madvillainy by Madvillain, an ensemble composed of MF Doom and Madlib, includes a song, entitled, *Curls* and the following lines are from said song with a CMU phonetic gloss.

*Villain get the money like **curls***
V IH1 L AH0 N, G EH1 T, DH AH1, M AH1 N IY0, L AY1 K, **K ER1 L Z**

*They just trying to get a nut like **squirrels** in his mad world*
DH EY1, JH AH1 S T, T R AY1 NG, T UW1, G EH1 T, EY1, N AH1 T, L AY1 K, **S K W ER1 AH0 L Z**, IH1 N, HH IH1 Z, M AE1 D, W ER1 L D

*Land of milk and honey with the **swirls***
L AE1 N D, AH1 V, M IH1 L K, AE1 N D, HH AH1 N IY0, W IH1 DH, DH AH1, **S W ER1 L Z**

*Where reckless naked girls get necklaces and **pearls***
W EH1 R, R EH1 K L AH0 S, N EY1 K AH0 D, G ER1 L Z, G EH1 T, N EH1 K L AH0 S IH0 Z, AE1 N D, **P ER1 L Z**

Consonance rhyming is present among '*curls*', '*squirrels*', '*swirls*', and '*pearls*'; where these words share the '**L Z'** consonant ending sound. The snippet '*money like curls*' is a multisyllabic rhyme, wherein there is a rhyme longer than one syllable; in addition, the phrases '*nut like squirrels*' and '*honey like swirls*' are also a multisyllabic rhyme. MF Doom is well-known for his flow because he delivers in a manner where everything in the set-up bar rhymes with the punchline bar. The last line is important to note because it is a holorime. A holorime is a rhyme where the entire line is rhymed with and this is evident in '*reckless naked girls get necklaces and pearls*'. It is important to note how MF Doom, as an MC, rhymes each word with the beat despite being inaudible upon reading his lyrics.

Aside from the aforementioned diverse set of rhymes in rap are techniques some MC's utilise to complicate their flow, they are crossing the bar line and motive. Crossing the bar line happens when the rhyme goes beyond the bar of four beats but still remains in the same line. MC's like Rakim and MF Doom often do this in an elaborate manner. Motive, on the other hand, is a short musical idea, musical fragment, or a succession of notes that has some special importance in a composition like *da-da-da-dum* in Beethoven's most famous symphony.[3] Kendrick Lamar often uses motives in his raps; his lines are often too quick and short to physically rap. The different types of rhymes present in rap and the diverse techniques are what make hip-hop music unique among other genres. Hip-hop is unlike any other genre and it strongly influences a variety of, if not all, popular genres today.

[3] https://www.vox.com/2016/5/19/11701976/rapping-deconstructed-best-rhymers-of-all-time

Corpus Construction

I. Format and Size

The corpus holds a total of eleven million words and more than a million and a half lines; it was formatted as a plain text file. The corpus contains lyrics from the most current and popular hip-hop artists at this point in time, such as Cardi B, Migos, Mac Miller, and Ugly God. However, it also includes artists dating back to the beginning of the hip-hop culture and movement—The Furious Five, Big Daddy Kane, and Dr.Dre, to name a few. Other smaller corpora were sourced from this corpus for testing and they include, in chronological order of when they first peaked in fame: Rakim, Jay-Z, and Kendrick Lamar. The lyrics, across all corpora included labels and metadata pertaining to each song, the labels are the following: *Artist*, *Album*, *Song*, and *Typed by*. Other metadata included chorus and verse markers. Here are some examples pulled from the corpus without metadata (the song titles and artists' names are provided for reference).

Dear Mr. President
the world has gone astray
brothers are dyin
they won't live to see today
was it all worth it
you had to lie to get your way
bloods thicker than water
what a price we had to pay

(*Dear Mr.President*, Fredwreck)

now, I'm adrock and I shock and I tick and I tock
and I can't stop with the body rhymes see
I got heart like john hittin mad starts
pass me the bat and I'll be rockin the whole park

(*Get It Together*, Beastie Boys)

Checkin' for the sisters
Flossin' the Prada
Dolce Gabana baby can I holla?
You know I'm peepin' at'cha
Hoping I can catch ya
I wanna get at'cha
Can we dance before the night is over?
Come on, oh

(Last Call, Xzibit)

FIGURE 2: Several excerpts extracted from the rap lyric corpus.

These excerpts present a microscopic lens into the format and content of the raw corpus. The following sections discuss the corpus' beginning and phonetic transformation.

II. Creating the Corpus

Gathering the corpus

The corpus was gathered in several steps, from harvesting initial links to extracting the text embedded in each link. The corpus was extracted from the site *ohhla.com*, an online database which houses a tremendous amount of Hip-Hop and Rap lyrics.

The website was crawled using three different python scripts for the sake of organisation. BeautifulSoup and Request, both python modules, were used to attain and parse the site's content using the html library parser. The entirety of the website's database was crawled with my scripts, including artists A through Z inclusive, for one expansive corpus.

The first script gathered all the initial sites needed for each artist. This script implemented a format where all the artists' names were put into a *csv* file under the header *artist* with a coupling column entitled *link*, and each link held all the links corresponding to that artist. Each link pertaining to an artist could have held a plethora of links, which was dealt with in the second step.

The second script crawled the links of each artist that the previous script wrote to a *csv* file, under the column *link*. Each link held a number of text links, some initial links lead to expansive discographies, but this pertained to certain artists. Some of the artists holding more extensive lyrical contributions to this corpus include, but are not limited to: Jay-Z, Nate Dogg, Rakim, Lil Wayne, Tupac, Nicki Minaj, and many more. Creating the links corpus was achieved by extracting all the embedded html links in each link of the csv file. Afterwards, the embedded link was cleaned and filtered using this regular expression '\.*txt$*'. This was done to find all links with this exact extension because the '.*txt*' ending links held the desired text for the corpus. After which these links were then written to a text file that held all found links. This text file entitled, '*link_corpus*', was eventually scrapped for text in the subsequent script.

To interject, an additional script to the three used to scrape *ohhla.com* was implemented once doubles were found in both the link and text corpus. This additional script is a middle man between scripts two and three, wherein, it removes duplicate links found in the text file *link_corpus*−created in the preceding script. The amount of initial links to be scrapped for text was 59,028 links, however, after removing the duplicates only 26,234 links were left to be scrapped in the following script.

The execution of the third script is when the raw unedited corpus begins to take shape. This last scrapper extracts all the text from the link corpus. However, difficulty was encountered in this step because certain links returned a *noneType* when parsing the html page for text; this occurred because the text in the tag that was being extracted didn't exist. The usage of a *try* and *except* clause allowed the deflection of a return error for certain links and wrote the text of each link to what would become the first raw corpus. Each song was preceded by the following metadata: artist, album, song, and typed by. The original script had written everything after the metadata in order to avoid useless information for each song

in the first version of the corpus and this was accomplished with a simple conditional. Although, after some advice to better analyze the corpus in later stages the conditional was removed because later work would require the knowledge of who wrote which songs. The cleaned corpus holds a grand total 11,107,365 words and 1,620,042 lines exactly, not including: metadata, dialogue, song cues, etc..

Website Structure, Content, and Crawling into the Complete Corpus

Putting the corpus together was no easy feat. For instance, assembling the link corpus posed a challenge because the links had to be extracted in different steps. While looping through the links, they were first filtered using a *find_all*, a function in *Beautifulsoup*, for all anchor tags ('a'). Afterwards, the hypertext references ('href') were extracted with the *get* function, wherein it was assigned to the variable 'url'. Lastly, in order to get the correct links, ending in '*.txt*', the regular expression weeded out the undesirable links from the 'url' variable. This resulted in incomplete links that ended in '*.txt*', which were then attached back to '*http://ohhla.com/*' and the outcome looked something like this:

http://ohhla.com/anonymous/ARTIST_NAME/ALBUM_TITLE/SONG_NAME.ARTIST_ACRONYM.txt

The found links would then be written to the link corpus plain text file and this same file would be crawled for text.

Another puzzling problem was extracting the text from links in the link corpus in the third script. At first the links were parsed similarly to what occurred in the previous scripts, where each link used request, a python module, to get the text in the link and then ran through *Beautifulsoup* for html parsing. However, when trying to return the value in a specific tag ('pre') holding pure text (lyrics) an error arose because some links were broken, thus resulting in a return Nonetype–breaking the code. This had been remedied with a try and except clause. This quick fix solution allowed for all non-empty links with the 'pre' tag to be parsed without halting the program unexpectedly. Moreover, the parsable links underwent a conditional removing metadata, which described the lyrics beforehand, such as: *Artists Name*, *Song Title*, *Album Name*, etc.. The by-product was the first iteration of the corpus.

III. Phonetic Transformation

Phoneticization Goals

The goal for the phonetic transformation was a python-readable phonetically transcribed corpus that included stress down to the word level. This included words not found in a traditional dictionary, seeing as rap can include ad-libs, such as, an emphatic '*yeaaaahh*' and other colloquial slang terms between and within verses. Even including non-traditional orthography for spellings of words, for example, 'gonna' versing 'going'. Because of the varying degrees of spelling and vocabulary, the phoneticization was achieved with the Carnegie Mellon University (CMU) pronunciation dictionary in conjunction with the Sequence-to-Sequence Grapheme-to-Phoneme (G2P) toolkit. The two combined successfully transcribed a large majority of the corpus.

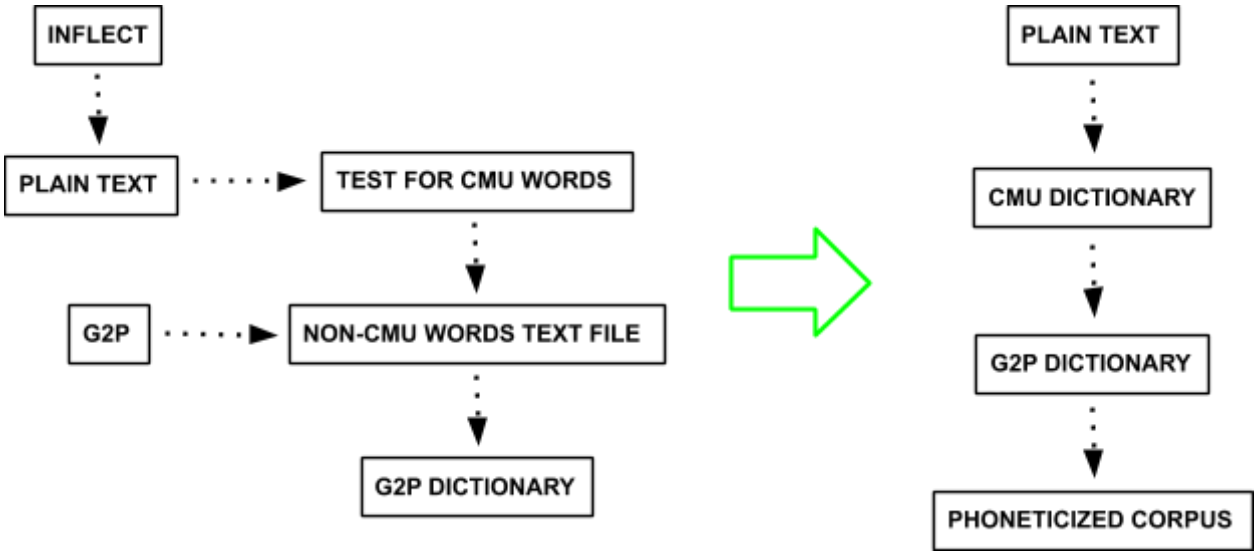

FIGURE 1: The process of phoneticization using inflect for the numerals and both G2P and CMU pronouncing dictionaries.

Tools and Output Format

The corpus was transformed into its phonetic counterpart with the CMU pronunciation dictionary using a python script that employed the dictionary's pythonic library entitled *Pronouncing*. The output is in *ARPABET* phonetic notation including stress for every word and it is formatted as a plain text file. In each line, every word is separated by a comma and each sound for that word is separated with a white space. The corpus was transcribed in a way that each *ARPABET* line is followed by its alphabetical equivalent.

ORIGINAL TEXT	You're a big man now
CMU-ARPABET NOTATION	Y UH1 R, AH0, B IH1 G, M AE1 N, N AW1

TABLE 1: A comparison between plain text and ARPABET notation.

This phonetic transcriber had been used because of the inclusion of stress in the output, which plays a part in rap. Stress is crucial in this study of rap lyrics because machine learning algorithms cannot detect the beat that goes with each song, so another way to measure the tempo or emphasis within a line is through stress.

What is *ARPABET*?

ARPABET is ASCII-phonetic notation including one to two letters per sound with stress markers, in numeric values, attached after the sound; 0 represents no-stress, 1 represents stress, and the rare 2 is secondary stress. In *ARPABET* notation, consonants are depicted with one letter, whereas vowels are embodied using two letters (most often two vowels) followed by the stress value. Furthermore, fricatives (θ, ʃ, ð) and nasals (ŋ) are portrayed using two letters, unlike other consonant sounds in the phonetic alphabet when transcribed into *ARPABET*. This form of phonetic glossing is useful for amalgamation with computer programs because of its readability and dodging of special non-ASCII characters.

Sequence-to-Sequence Grapheme-to-Phoneme (G2P) Toolkit

The python toolkit, Sequence-to-Sequence G2P, was used for the words not present in the CMU pronunciation dictionary, which is resourceful, as a phonetic tool, but far from perfect as it does not hold all existing words and their pronunciations, or possible variants. The corpus is over a million words and the CMU dictionary was unable to transcribe fifty-thousand words. This was discovered through a simple loop which filtered through the output and when an empty list was returned it wrote the word to a separate file that documented the untranscribable words. This file was filtered again to remove the duplicates leaving a total of fifty-thousand words. These words represent the missing words in the CMU dictionary, they include numbers, ad-libs, artists' names, and word variations, such as walkin' as opposed to walking.

The transcription inconsistencies of numerous lines called for a supplemental phonetizer. This tool does Grapheme-to-Phoneme conversion using a transformer model from the *tensor-2-tensor* toolkit (Cmusphinx). This toolkit was used to create a separate dictionary for the words untranscribed by the CMU dictionary entitled, '*Supplementary Dictionary*' in the format of a plain text file. The corpus was then transcribed again using the two dictionaries, ensuring a phonetic counterpart for all words in the corpus. During the second transcription, however, when the CMU dictionary would return an empty value (because it could not transcribe an unknown word) the script would call the Supplementary Dictionary for that word and in the end a mostly phoneticized corpus was produced.

Despite producing a fully phoneticized corpus with the Supplementary Dictionary, there is a drawback to the G2P toolkit, which is its lack of stress implementation. This is important to note as fifty-thousand words would be missing stress, meaning that 43% of the corpus would be missing stress however this is looked over because the phonetic transcription is more important than the stress notation. To conclude, an ASCII phoneticized corpus was accomplished.

Difficulties

Prior to the success of the CMU pronunciation dictionary, was the mistake of using EPITRAN, an IPA phonetic transcriber. EPITRAN transformed the text using IPA notation and did not include stress. The inclusion of IPA characters was problematic because these special characters are not easily readable by computer programming languages, such as python. Thus, this transcriber needed to be transformed into SAMPA, which does not use IPA symbols, instead it uses punctuation and upper-case letters in place of IPA symbols. SAMPA is just an ASCII-alternative to IPA. A script to reformat the EPITRAN output to SAMPA output had been executed but the lack of stress in each word's pronunciation ruled out this option. In addition, the ASCII transformed EPITRAN is seen as a weak conduit because of the difficulty when reading–rendering it cipher-like.

For example, take a line from the corpus transcribed with EPITRAN, converted to SAMPA characters and compare it to the CMU transcription that separates words by commas.

ORIGINAL TEXT	Promised never to cry
EPITRAN PHONETICIZATION	pɹɑməst nɛvɹ ˌ tə kɹaj
ASCII-EPITRAN PHONETICIZATION	prAm@st nEvR t@ kraj
CMU-ARPABET PHONETICIZATION	P R AA1 M AH0 S T, N EH1 V ER0, T UW1, K R AY1

TABLE 2: A comparison of different phonetic outputs

The CMU text is intuitively easier to read because it employs purely alphabetical letters aside from stress notation, whereas, the EPITRAN text, in both formats, contains special characters and these characters are less obvious when reading. After comparing the two options, it is evident which of the two is superior, in terms of quality and stress implementation—the CMU pronunciation dictionary.

Another conquered difficulty, involving phoneticization, was the installation of the Sequence-to-Sequence G2P toolkit; it required Google's Tensorflow and Tensor-2-Tensor, the programs required lots of space, troubleshooting, time, and patience. The problems included when installing the toolkit were the python version, the necessity of other modules, and the correct version of Tensor-2-Tensor. The proper installation of the toolkit came with implementing the correct versions of several modules and putting it on a separate server all-together because of its lengthy process. After, it was placed into a python virtual environment with select python modules in certain versions. Once the toolkit was properly installed, the Supplemental Dictionary was created for a mostly transcribed corpus.

One undealt challenge was the phoneticization of numeric values and numeric-alphabetical values. These include simple numbers, dates and years, phone numbers, and numeronyms.

Numerical Values	2	0-4	757	4686605
Dates and Years	‘93	2000	1970s	Y-2-k
Phone Numbers	2-1-3	702-386-5397	1-800-roadrage	9-1-1
Numeronyms	106th	E-40	18-year-old	300k

TABLE 3: A table holding some non-transcribed numerical values.

Most numericals and numeronyms were transformed using inflect; inflect is a python module that takes numerical values and returns their alphabetical counterpart. Their transformation allowed for them to be transcribed phonetically. With the exception of special cases almost all values were transformed; only two thousand strings remain untranscribed in the entire corpus of eleven million words. This is why the corpus is mostly transcribed–not fully.

Related Work

The following study discusses the automatic generation of rap. The model and methods are presented and act as a blueprint and catalyst for the methods in this study.

I. Deepbeat, Rap Lyric Generator

The academic paper entitled *DopeLearning: A computational Approach to Rap Lyric Generation* was interested in the creative, meaningful production of rap lyrics. The study breaks down the computational approach to generating rap lyrics with a prediction model that found the next line with two different models. The first used RankSVM algorithms and the second used a deep neural network. The RankSVM algorithms were intended to combine different relevance features for the next-line prediction problem, the baseline for their rap-lyrics generator; the neural network model was designed to match the semantic similarity between the query line and the hereinafter possible lines (Malmi et al., 2016).

Their study highlighted the problems with computational creativity involved in producing rap lyrics whereas the results merged human interaction with a text generation machine that was creative. This field of study intrigued the researchers because of the complexity involved in rap lyrics. In their study, they tested the rhyme density on a word by word basis for the most popular rappers in order to test the quality of rhymes. To find the next line, the lyric generator took an information retrieval (IR) approach which was meant to identify the most relevant line of all the candidate lines with respect to the query. Information retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) (Manning, Raghavan, & Schütze 2008). The information being retrieved in this study was from a set of lines in the corpus that they gathered from *ohhla.com*. However, their corpus was only composed of 104 popular English-speaking rap artist.

The use of *combinatorial creativity* produced innovative novel combinations of familiar ideas as an outcome; this form of creativity has been used to create poetry as well. The neural network architecture they implemented was a feed-forward net, which is the most optimal for short sentences, like lyrics in this case. In combination with the aforementioned, the study utilized the bag-of-words method to push for more semantically coherent results, or a relevant next line; this was just one of the four methods used for coherence. With the bag-of-words they executed the Jaccard measure of similarity between the bag-of-words between the query line and one of the possible options (Malmi et al., 2016).The other methods for finding optimal semantic coherence were *BOW5*, which analyzed other potential lines at the same time (Malmi et al., 2016). A latent semantic analysis was used as well for synonymy and polysemy; the last semantic feature was based on a neural language model, also referred to as NN5. The NN5 method first started by finding distributed vector representations for words and then based on this vector representation of text, their network learned to predict whether it believed that the last line was a suitable next line candidate or not (Malmi et al., 2016). Furthermore, their RankSVM algorithm provided a relevance score of each line based on three features: rhymes, structure, and semantic features—in order to

determine the next line based on the difference between the query and the list of possibilities. Both the NN5 and the RankSVM algorithms learned on a training set and became successful in choosing the next rap lyric line on the human preference level.

In addition to their study, they created an online rap lyric generator called *Deepbeat* (deepbeat.org), which includes a deep-learning version. The deep-learning option was intended to provide more semantically alike lines and denser rhymes; it also learns from the user's usage logs and produces relevant lines with the user's preference in mind. The online interface generates lyrics using optional thematic keywords from the user's input or the user's choice of a line provided. The lines produced are based on the user's input and the lines following can be modified by an optional thematic keyword. The generated lyrics, or song, can be saved after the user is satisfied. The lines produced are sourced from the throng of artists present in their corpus and each artists' line is identified with the corresponding album cover. The following lines are lyrics produced using the *Deepbeat* generator–making use of the deep-learning option. The keyword *honey* along with the accompanying line *melts like butter and taste like money, honey* are both user-input. Both were personally chosen and used as input for their *Deepbeat* rap lyric generator.

melts like butter and taste like money, honey (A) ← user-input line
Ha ha ha ha ha make money money money (A)
Because we be Husslin Husslin Husslin Husslin.. (B)
And if I'm at a jam it must be pumping (C)
WE GOT dangerous hang out spots and slick cops (D)
She was a brown skin ebony honey that love hip hop (D)
She got the face that you wanna spend money on (E)
You don't measure a man by what he got (D)
Followed by three more boys and then a baby girl (F)

FIGURE 4: Output from the *Deepbeat* rap lyric generator.

The rhymes were analyzed against the user-input line. There is an array of rhymes present in these nine lines produced by *Deepbeat*; they are: alliteration, assonance, consonance, internal, multisyllabic, perfect, and single-syllable. The semantic coherence among the lines is evident thanks to the fours methods they chose to solve this objective; the lines produced are relevant to the query line, notably the query line contains the word *money* and the lines *Because we be Husslin Husslin Husslin Husslin* and *You don't measure a man by what he got* both allude to *money*. The first line emanates the idea of earning money while the second discusses the notion of measuring someone's wealth; both lines do not directly include the word *money*, but are semantically coherent because they imply a fundamental pertinent to the concept of money—income. Other examples of semantic coherence are present in these lines, such as, the relation between ebony (line 6) and honey (line 1) both colors and the relation to children and money, seeing as that could be another measure for wealth. Furthermore, the techniques and algorithms used in this study have proven to execute on a capacity alike that of a human's ability to rhyme. However, this is probably attributed to the fact that the lines are not organically produced but automatically

coordinated to rhyme with one another. Their study overcomes the difficulty of studying rhymes in rap and how the analysis of rhymes could be used to create new songs from a corpus of already-existing rap lyrics.

Manual and Automatic Annotation of Rhyming Schemes

I. Why is manual annotation necessary?

The rap lyrics had to be studied manually before automatic production. This is because a machine would have absolutely no clue on how to produce rap lyrics, which rhyme, without clues or training beforehand. Manual annotation illuminates rules for rhyming in rap lyrics; it can present patterns and they can be implemented when generating rap lyrics. The following lines are an example of how manual annotation can be handled in one fashion.

TYPE	LINE	END SOUND
TEXT	more juice than *tropic**ana***	cana
PHONES	M AO1 R, JH UW1 S, DH AE1 N, *T R AA2 P IH0 K **AE1 N AH0***	AE1 N AH0
TEXT	rock from New York to *Sav**annah***	annah
PHONES	R AA1 K, F R AH1 M, NUW1, Y AO1 R K, T UW1, *S AH0 V **AE1 N AH0***	AE1 N AH0

TABLE 4: An example of manual annotation.

Upon first glance of the lines' phonetic counterparts, it is evident that '*tropicana*' and '*Savannah*' share a perfect rhyme through stress, but more importantly through matching end sounds. Manual analysis, like the one done in *table four* presents patterns, such as, the '*AE1 N AH0*' (ana-annah) sound and pattern; this type of analysis can be useful in different ways, for example, using the pattern above would make it possible to find all other lines that end in the '*AE1 N AH0*' sound pattern, wherein, these lines could be classified according to this pattern. This same process could be replicated across the entire corpus in order to find all the existing patterns, creating a possibility for all lines to be classified under different patterns. After which, a user could decide on a pattern and produce a new song from the vast amount of lyrics in the rap corpus. Lyrics innately hold rhyming patterns and once these patterns are realized after close inspection of the phones, it is possible to mimic them with automatic generation; thus, elaborating as to why manual annotation is significant.

II. How to do it manually: where to look for rhymes?

Manual annotation was a process of different steps. The first involved answering certain questions and the second being the construction of the lines based on them. The questions were:

- How many lines should be analysed?
- Should the lines be analysed in pairs? Or verses?
- Where should the lines be taken from specifically in the corpus?
- What should be the criteria for the rhymes ?
- Should the whole line be analyzed? Or just the last word? Or just certain parts?

A Hundred Lines

The manual annotation was an analysis of ten randomly selected sets of lines each holding ten lines adding to a total of a hundred lines. For each line in the set, there was a plain text line and its phonetic counterpart line. To fully grasp the rhyme complexities—in the genre, the lines were analysed for assonance, consonance, perfect, and slant rhymes, or a combination of these said rhymes. The final word in each line was compared to the final word of all preceding lines for the aforementioned rhymes. Additionally, the presence of alliteration and multisyllabic rhymes were analyzed throughout the lines. The following table exhibits the manner in which a set of lines were manually annotated.

ID	TEXT/PHONES	RHYMING LINE	RHYME TYPE
1	Face down don't turn around	—	
	F EY1 S, D AW1 N, D OW1 N T, T ER1 N, ER0 AW1 N D		
2	listen to the bass pound	1	PERFECT
	L IH1 S AH0 N, T UW1, DH AH0, B AE1 S, P AW1 N D		
3	For all of the sound	1	PERFECT
	F R AH1 M, AO1 L, AH1 V, DH AH0, S AW1 N D		
4	this ain't no star spangled banner	—	
	DH IH1 S, EY1 N T, N OW1, S T AA1 R, S P AE1 NG G AH0 L D, B AE1 N ER0		
5	more juice than tropicana	—	
	M AO1 R, JH UW1 S, DH AE1 N, T R AA2 P IH0 K AE1 N AH0		
6	Rock from New York to Savannah	5	PERFECT
	R AA1 K, F R AH1 M, N UW1, Y AO1 R K, T UW1, S AH0 V AE1 N AH0		
7	Cause it's Macka Framma?	—	
	K AA1 Z, IH1 T S, M AE K AH, F R AE M AH		
8	When I go to a show	—	
	W EH1 N, AY1, G OW1, T UW1, AH0, SH OW1		
9	Some run some come plenty have fun	—	
	S AH1 M, R AH1 N, S AH1 M, K AH1 M, P L EH1 N T IY0, HH AE1 V, F AH1 N		
10	To watch a nig get dumb	9	SLANT
	T UW1, W AA1 CH, AH0, G EH1 T, D AH1 M		

TABLE 5: A set of ten lines manually annotated.

Referring back to the questions put forward, the lines were analyzed through their final word, more specifically the last sound in end word. Each end word was compared to its preceding lines. If there was a rhyme match then the line would be given the ID value of the first instance of the found rhyme in the set. So, in particular, lines two and three both hold the value of one in the rhyming line column because line one first presented the ending pattern of 'AW1 N D'; they are labelled perfect, in the rhyming type column. The same goes for the other found rhymes; the perfect in lines five and six and the slant in lines nine and ten .

The role of alliteration and multisyllabic rhymes

The lines were also manually annotated for alliteration and multisyllabic rhymes in a separate column after the first four initial columns. Alliteration was tested on the sound of a word found in the phonetic text, not the original text. In particular, if a word began with a 'T' but was followed by an 'H', thus, creating the fricative 'TH' sound it would not share alliterative features with another word beginning with a non-fricative 'T'. To illustrate the phrase 'that town' would share no alliterative feature, but *those thoughts* and *today's tomorrow* both possess alliteration. The role of alliteration is important in rhyming because it is added rhyming to a line; it can be thought of as bringing the element of musicality, or musical rhythms, to rap lyrics.

III. How to automatize: how to reproduce manual annotation?

The first step producing lyrics automatically was the automation of annotating them. This was done through a program that replicated the same process of manual annotation. The process of automatization analyzed each set just like the manual annotation. The automated component took in the same file holding the manual annotations, however, the only values accounted for were the ID, original text, and phonetic text. The rest was pushed after the automatic output so its values can be compared to the manual ones. The automatic analysis searched for the same rhymes, but if assonance or consonance were found the return output was assonance or consonance *slant*. This program was first run on the manually annotated lined and then on a larger set after it was perfected.

Five hundred lines

The same program was used on a larger set of random lines, this time fifty sets–adding up to a total of five hundred lines. This larger collection was not manually annotated, instead these lines were used for verifying that the rhymes in the current and smaller collection had been analysed correctly. The results proved positive for this collection of five hundred lines since the output was accurate after inspection. Moreover, this analysis was also useful in finding false positives and false negatives of found rhymes.

False Positives—False Negatives

These sets were great for analyzing false positives and false negatives. The found false positives included words ending in *-le*, such as in: crumble, brittle, and dwindle. Another false positive came with words ending in *-ing*, for instance: crying, blowing, and riding. These are examples of found false positives because the words *crumble* and *brittle* don't necessarily rhyme, however, do share an end sound. The same goes for *-ing* ending

words. For these false positives an additional rule was implemented wherein the sound before *-ing* and *-le* was also analyzed when looking for rhyming matches. For example, *riding* does not rhyme with *crying* nor does *bumble* with *brittle*, but *crying* does rhyme with *trying* and *bumble* rhymes with *rumble*. The implementation of this rule would exude more accurate rhymes among matches that share these common word endings.

What is Word2Vec?

Word2vec is the idea of inferring knowledge on a word based on the surrounding context. In essence, it is the neighbors of a word (the target word) that will represent the information about this target word's meaning. This tool, word2vec is useful for finding an array of similar words related to the target word; it can even find dissimilar words. This idea of word representation is also known as word embeddings. The word2vec implementation used in this study came from *Gensim*, an open source python library for natural language processing, with a focus on topic modeling. In *Gensim*, word2vec is an algorithm for learning about word embeddings from a corpus. Word2vec was used in this study for determining an appropriate window size, the word embeddings, and for producing verses.

I. Simple Word2Vec Training on the Corpus

The corpus was trained on a multitude of window sizes; window sizes are the extension of tokens from the target word. For instance, if there is a phrase '*the green and yellow happy toad*' with a window size of two and the target word is *green*. This means that the meaning for the word *green* will be based on '*the*' (backwards) and '*and yellow*' (forward). The corpus was trained on different windows sizes. In the next subsection there is a discussion of window sizes: five, fifteen, and thirty with each size emitting different results for an input query.

Window Sizes

The number of windows given when training a model changes the embedding of any target word. Take the phrase mentioned above, if the window size were reduced to just one, then its neighbors would be '*the*' and '*and*'. Of course when performing these tests, it is important to eliminate common words like: *the*, *and*, *a*, *of*, etc.. The elimination of common words, or stop words, is important because these words carry no semantic importance nor important information about the target word. So, in reality, with a window of one and the target word set to '*green*', this means its neighbors would be just '*yellow*' (forward), not including any stop words. The words: *tangerine*, *ocean*, and *harmony* were used when evaluating and testing different window sizes.

TANGERINE			
SIMILAR		DISSIMILAR	
WORD	SCORE	WORD	SCORE
WINDOW SIZE 5			
Raspberry Mango Strawberry Okra Turquoise	0.6697152853012085 0.6620488166809082 0.6545091867446899 0.6500412225723267 0.6455582976341248	Stressin Followin Upstate Hatin Helpin	0.38966745138168335 0.35303181409835815 0.32514840364456177 0.3245125412940979 0.3081620931625366

WINDOW SIZE 15			
Mango	0.5456982254981995	Clout	0.2552225589752197
Strawberry	0.5326031446456909	Stressin	0.2525743246078491
Turquoise	0.5080513954162598	Accept	0.2391214668750763
Oatmeal	0.5006126165390015	Maintain	0.2378636747598648
Cran	0.4853413701057434	Questions	0.2353099286556244
WINDOW SIZE 30			
Mango	0.4455430507659912	Understand	0.2505989670753479
Cran	0.4203491508960724	Munnies	0.22149497270584106
Lavender	0.4082515239715576	Sayin	0.2131597399711609
Scrimps	0.4076573848724365	Bec	0.2094455361366272
Raspberry	0.4050835371017456	Wrong	0.20111098885536194

TABLE 6: The effects of different window sizes for similar and dissimilar words to the word *tangerine*.

The incrementation of window sizes for this term is interesting because the window size of five returns the word *turquoise* as oddly similar to the word *tangerine* and this similarity is increased with a window size of fifteen, however absent when the window is set to thirty. It could perhaps be said that the word, *tangerine*, in a window size of five and ten is also interpreted as a color seeing how it is scored highly similar to *turquoise*.

On the other hand when the window size is set to thirty, the five most similar words are all food and plant related objects, of course, the word *tangerine* could also be seen as a color as well so deciding which window is best for this term is subjective. However in cases where the corpus is expansive–a million-plus words–a larger window size is seen as beneficial. As for the dissimilar terms, their distance to the query word is lengthened with a larger window size, such as these words: *wrong*, *questions*, *accept*, etc..

HARMONY			
SIMILAR		DISSIMILAR	
WORD	SCORE	WORD	SCORE
WINDOW SIZE 5			
Intertwined	0.55000901222229	Fired	0.3824250102043152
Infamy	0.5336099863052368	Interviewin	0.3356168866157532
Equality	0.4841376543045044	Shut	0.3350527286529541
Cosmos	0.47860825061798096	Outta	0.3184851408004761
Actuality	0.460758239030838	Gassed	0.31208062171936035
WINDOW SIZE 15			

Intertwined	0.413132905960083	Outta	0.28010088205337524
Providing	0.39210787415504456	Gassed	0.2534438967704773
Encrypted	0.37748250365257263	Jacked	0.247830331325531
Eyay	0.37148797512054443	Fired	0.23943692445755005
Devotion	0.36627212166786194	Confirming	0.22208787500858307
WINDOW SIZE 30			
Intertwined	0.3193565905094147	Tooken	0.23952272534370422
Mystical	0.31689465045928955	Poley	0.21026742458343506
Tayyyy	0.3110637068748474	Ped	0.20524857938289642
Portraits	0.3087109923362732	Looka	0.2028065025806427
Unity	0.3067011833190918	Goddamn	0.2011372298002243

TABLE 7: Various window sizes for related and unrelated terms to the word *harmony*.

The output for *harmony* is from a window of five to a window of thirty and shows some interesting transitions. For example, the term *unity* is synonymous with *harmony* however, present only when the window is extended to thirty. But on the other hand the term *Tayyyy* (presumably a name) is seen as similar when the window is extended to a value of thirty. Even the dissimilar terms in the window of thirty have some weird outputs subjectively. Perhaps, it can be said that a window around fifteen is a good match for the corpus, which is a good medium for a larger corpus.

OCEAN			
SIMILAR		DISSIMILAR	
WORD	SCORE	WORD	SCORE
WINDOW SIZE 5			
Sand	0.6572503447532654	Cockers	0.3700949549674988
Sea	0.645756185054779	Disrespectin	0.34067901968955994
Lake	0.6439104676246643	Earned	0.339028537273407
Shore	0.6063495874404907	Lackin	0.33240872621536255
Pond	0.5981467962265015	Testin	0.3227313160896301
WINDOW SIZE 15			
Sea	0.5844480395317078	Mafuckas	0.2695673406124115
Lake	0.525606632232666	Rea	0.2687361240386963
Oceans	0.4892445206642151	Wantin	0.2609642744064331
River	0.4685331881046295	Contributing	0.25153040885925293
Floating	0.46256887912750244	Incase	0.2440931499004364
WINDOW SIZE 30			
Sea	0.4739564061164856	Gutless	0.2172885239124298
Oceans	0.42214512825012207	Rea	0.2133774757385254

Waters	0.39863330125808716	Polluters	0.2055002748966217
Floating	0.3788316249847412	Swallowers	0.20292310416698456
Water	0.37452518939971924	Contributing	0.199842169880867

TABLE 8: Insight to different windows for related and unrelated words to the query *ocean*.

Across Table 8, there is evidence of variation among the related and unrelated terms. In the section dedicated to similar words to ocean, only words directly related to water are mentioned in a window size of thirty. On the other hand, when the window size is set to five the related terms are less unified seeing as some terms represent other types of bodies of water, such as, *pond* and *lake*. There is an evident transition among the related terms when going from a window of five to one of thirty. In regards to the unrelated terms, they simply get more and more unrelated as the window size is increased.

The Model

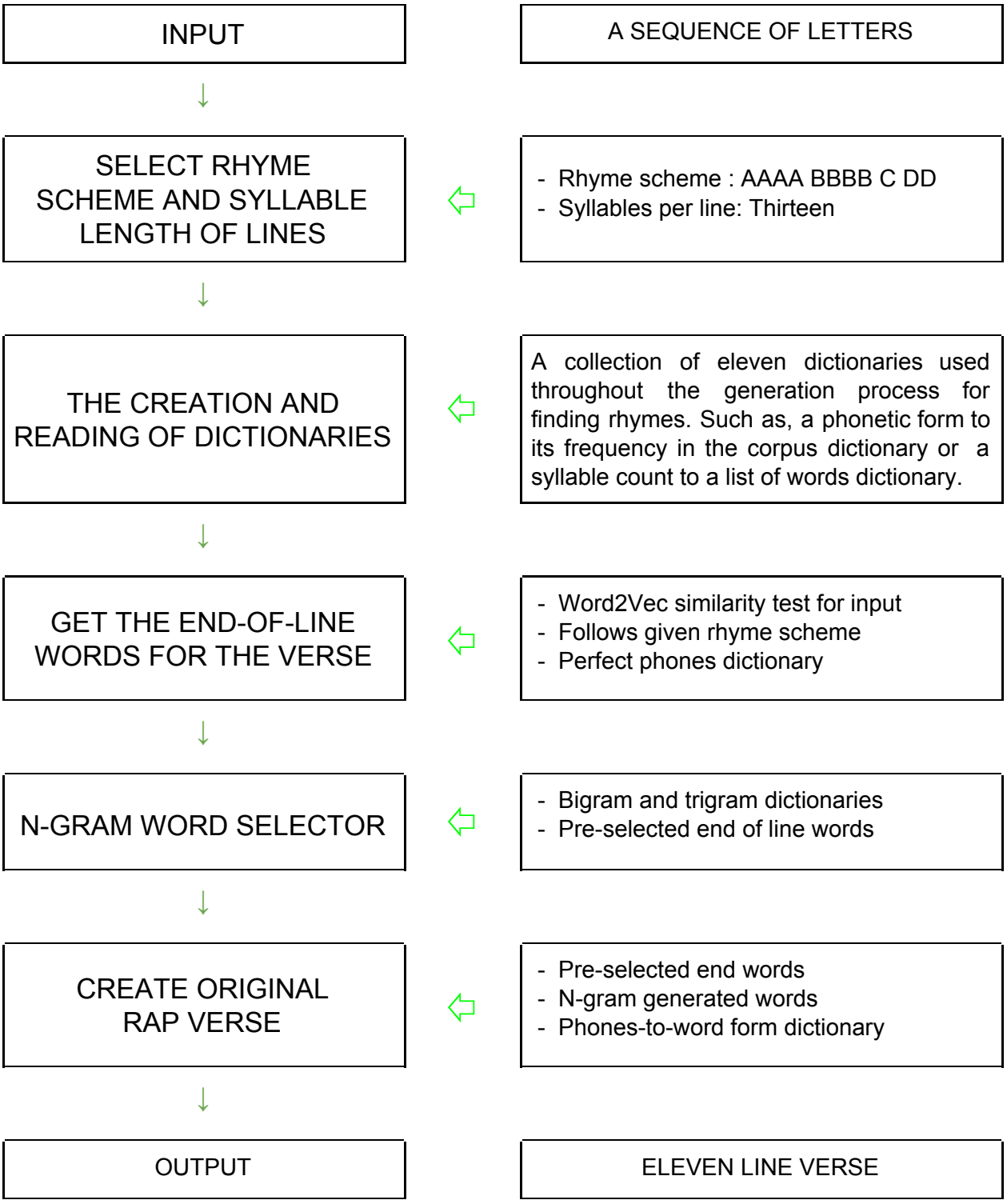

FIGURE 5: An overview of the process involved for generating automatic rap lyrics.

Rap Generation

Moving forward, the rap generated in this study had several constraints. The first being that each line, well bar, generated had to contain thirteen syllables, a medium between the minimum and maximum average of syllables in a bar of rap. Another constraint implemented during the generation was a rhyme scheme, which is: AAAA BBBB C DD. The following subsections are a walk-through of the model and how original rap lyrics were generated.

I. Training a Model on the Corpus, Pre-Rap Generator

The pure text corpus was prepped by transforming the corpus into tokenized sentences. This important training on the corpus was done with *Gensim* and its *word2vec* algorithm. The training on the corpus was useful for finding words similar to the input, just as what was done for tables six through eight. But in the case of rap generation, the number of similar words returned when given an input was increased to one thousand. First, these similarly generated words would be used as options for the end-of-line words in accordance with the rhyme scheme. Second, they would be used as options to fill in the lines with the n-gram word selector. Furthermore, the input could have been a word, a pair of words, a phrase, etc..

II. Dictionaries

Before the process of generation, comes the creation of eleven dictionaries; each plays an important role for the rap generator to produce original lyrics. These dictionaries are used after the creation of the *Word2Vec* model. The returned dictionaries are utilized in the various functions needed to generate a verse. The following is a breakdown of all the dictionaries.

N-Gram Dictionaries

There were two n-gram dictionaries created, bigram and trigram dictionary, wherein each of the found n-grams were accompanied by their frequencies in the corpus.

A bigram between *stop* and *glamour* with a frequency of two.		
S T AA1 P	G L AE1 M ER0	2

TABLE 9: An example pulled straight from the bigram dictionary.

A trigram with the words: *carrying*, *darkness*, and *the* sharing a frequency of 3.			
K AE1 R IY0 IH0 NG	D AA1 R K N AH0 S	DH AH0	3

TABLE 10: An example of a trigram taken directly from the trigram dictionary.

Both of these n-gram dictionaries play a crucial part in generating rap lyrics for the generator. In fact, these dictionaries are responsible for filling in the lines of the verse before the end-of-line word.

Syllable Frequency

This dictionary classifies the corpus' vocabulary by the number of syllables in each word, wherein the key is the syllable count and its values are all the corresponding phoneticized words. The first key started at one (syllable) and went as high as one hundred and twenty-nine (syllables). Furthermore, this dictionary was used for the generation of text preceding the end-of-line word. To illustrate, a line is pre-filled with the final word before using this dictionary. So, if the final word is three syllables then it would allow for all words less than ten syllables to be candidate fillers of this line. This is because only thirteen syllables are allowed, thus, we subtract the number of syllables in the final word, in this case three, and are left with ten available syllable spots to be filled. This dictionary aids in the filling of the verses' lines.

Phonetic Form Frequency Dictionary

In this dictionary, the phonetic form of the word is the key and its frequency across the corpus is the value. For example, the phonetic word HH OW M IY Z (homies) is present 2,046 times. This dictionary was used in various functions for randomly selecting the phonetic form. The frequency of these phones acted as the weight for selecting the phonetic form while using the random function which would return just one of the phonetic forms to be used in the generated verse.

Phonetic Form to Word Form Variants

This dictionary presents the different spellings as the values for a phonetic form, which is the key. For instance, this phonetic form and key, K R IH1 M AH0 N AH0 L has these spellings as its values: *CRIMINAL*, *Criminal*, *crim-inal*, *crimi-nal*, *crimin-al*, and *criminal*. Whereas this phonetic form, P IH G AH D IY (the key), has fewer values: *piggedy*, and *piggidy*. This dictionary is useful for knowing the variations of spellings for a word throughout the corpus while they share one phonetic form.

Phonetic Form Endings

For this dictionary, the ending of the phonetic form is extracted and repurposed as the value and the key is the fully intact phonetic form. By extracting the ends of the phonetic form, rhymes can be found much more easily, more specifically perfect rhymes. The values in this case would be the last vowel and consonant. If the word ends with a vowel, then an underscore is used as a placeholder for the nonexistent final consonant.

Phonetic Form (Key)	Ending Phones (Value)
IH2 M P EY1 L (impale)	EY1 L
IH2 N F ER1 N OW2 (inferno)	OW2 _

TABLE 11: An excerpt from the *Phonetic Word-Endings* dictionary.

Word-To-Phonetic Form Dictionary

This dictionary has taken the corpus and produced a dictionary in which the key is the original alphabetical text and the value is the key's phonetic counterpart.

Alphabetical Form (Key)	Phonetic Form (Value)
inhibit	IH2 N HH IH1 B AH0 T
toothpaste	T UW1 TH P EY2 S T

TABLE 12: A glimpse into the *Word-to-Phones* dictionary.

Word Frequency Dictionary

This is a simple dictionary, which uses the alphabetical form as the key and its frequency across the corpus as the value.

Alphabetical Form (Key)	Word Frequency (Value)
Alert	66
nutrition	7
rebuild	42

TABLE 13: A fragment of the *Word Frequency* Dictionary.

Phonetic Form Endings and Matching Possibilities

For this dictionary, a vowel sound is the key and its values are all the possible ends it can produce taken directly from the multitude of words present in the corpus.

Vowel (Key)	Possible Ends (Value)
IY	IY-B, IY-B Z, IY-CH, IY-CH T, IY-D, IY-D K, IY-D P R T, IY-D Z, IY-DH, IY-DH D, IY-DH Z, IY-F, IY-F R Z, IY-F S, IY-F T, IY-G, IY-G D, IY-G N, IY-G N Z, IY-G Z, IY-HH, IY-JH, IY-K, IY-K F, IY-K N D, IY-K S, IY-L T, IY-L TH, IY-L Z IY-M, IY-M D, IY-M S, IY-M T, IY-M Z, IY-N, IY-N L, IY-N M F, IY-N S, IY-N T, IY-N T S

TABLE 14: A snippet of the *Phone-Endings* dictionary with matches for the vowel-sound *IY*.

The Perfect Rhyming Dictionary

The phonetic processing of the corpus assigned a 0, 1, 2, or blank after each vowel as stress marker: a zero means no stress, one is for primary stress, two stands for secondary stress, and a blank means that it was a word phoneticized by the G2P Sequence-to-Sequence phoneticizer. Which did not supply stress during phoneticization for

nonexistent words in the CMU dictionary. So this dictionary levels out all the possible variants and makes them equal to one another. This is useful for finding *perfect* rhymes. If they were differentiated from one another then fewer perfect rhymes would be available. For instance, the end sound *AA N T S* has three variants present in the corpus: *AA0-N T S*, *AA1-N T S*, and *AA2-N T S*, where as, this end sound *OW F S* possesses just one variant *OW1-F S*. Notice the different stress markers after the vowels.

Phonetic Form Endings and Corresponding Words Dictionary

This collection of phonetic forms, the end sound is the key, such as, *AA B*, and the values are all the words possessing this word ending. The same function to find perfect rhymes in the section *Manual and Automatic Annotation of Rhyming Schemes* was used to create these groups of perfect rhymes.

Word Ending	Phonetic Form of Words Sharing this Word Ending
AA B	AA B, AA D B AA B, D R AA B, D UW K N AA B, EH M IY AA B, EH N Z IH R AA B, F EY S M AA B, F IH T AH S T M AA B, HH AA B, HH AE N JH AA B, HH AY N D JH AA B, IY AA B, JH AO HH AA B, K L AA B, R AA B, S AA B, S AY B AY D IY B AA B, S IY K AA B, S K W AA B, S M AA B, S P AH N JH B AA B, SH AA B, SH IH S K AH B AA B, SH IH SH K AH B AA B, Z AA B
IH B	AE B L IH B, AE D L IH B, AY IY AY IY IY IY IY IY IY IH B, D IH B, D IH B, D AE B D IH B, F IH B, G R IH B, JH EY L IH B, K AE N IH B, K AE R IH B, K L IH B, K R AY IH IY IY IH B, K R IH B, M AE D L IH B, M AE N F IH B, P IH B, R IH B, S AE HH IH B, S EH K T IH B, T AE L IH B, V IH B

TABLE 15: The end sounds *AA B* and *IH B* and the corresponding words with these endings.

The phones *SH IH SH K AH B AA B* represents shishkebab and the phones *M AE D L IH B* stands for Madlib, just some examples.

III. The Generator

The generator used various methods to produce original rap lyrics after creating a *word2vec* model and reading the aforementioned dictionaries. Each method worked with the constraint of thirteen syllables and the *AAAA BBBB C DD* rhyme scheme. Despite the fact that both were adjustable, the discussion of generation will center around these previously mentioned constraints. The process of generating lyrics for this study is referred to as a backwards n-gram model. This is because the final words are chosen first to fit the rhyme scheme, which is then followed by an n-gram implementation for filling in the lines prior to the final word. Furthermore, the generator was capable of producing any number of raps per run with each rap carrying unique end-of-line words for every verse.

Line-Final Words

The generator was first tasked with finding the possible end-of-line words based on the input. The input was used to extract one thousand similar words using the *word2vec* model. The model was trained on a window size of fifteen—a good medium among the tested windows based on tables six through eight. These similar words were then run through the *Word-to-Phonetic Form* dictionary, and were then filtered to remove any duplicates. Keeping only unique forms, helped in avoiding the appearance of duplicates in the returned verse. Once this was done, a new dictionary was created to match each of the one thousand similar words against the other similar words in order to find a perfect rhyme matches amongst these words directly related to the input. This new dictionary found perfect matches with the aid of the *perfect rhyming* dictionary.

Once this dictionary was established, the generator looped through this new dictionary to find potentials for each part of the rhyme scheme. The rhyme scheme was formatted into a list: 4, 4, 1, 2; this pattern represents *AAAA BBBB C DD*. Looping through each part of the rhyme scheme, allowed for groups of potential rhyming words to form, which were then filtered again to eliminate doubles. While using weights, groups of possible words were randomly chosen; the weights came from the values of the *Phonetic Form Frequency* dictionary—for the corresponding word. This was so a random selection was made according to the relative weights. After these potential groups were narrowed down to three or four options per line, came the usage of the *perfect rhyming* dictionary for a second round of filtration. This ensured that each segment of the verse was true to the rhyme scheme by possessing perfect rhymes unique to each segment.

The production of unique perfect rhymes in accordance with the rhyme scheme was given ten attempts in order to avoid producing duplicates for the end-of-lines words. This means the generator tried up to ten times for each segment of the rhyme scheme to provide unique end-of-line perfect rhyming words. Once this was the case, they were recognized as the final end-of-line words—no longer potentials.

The fulfillment of rhyming final words across the verse, allows for rhymes to be easily recognizable. It adds to the flow of the verse and also creates an automatic link among the lines through this shared perfect rhyme; hence, the reason why line-final words were found first. This is where the *backwards* part of the backwards n-gram model comes into play. In addition, choosing the line-final words with word vectors based on the input allowed for a direct and obvious link between the input and the verse generated. However, this method is just half of the backwards n-gram model used for rap generation.

N-Gram Application

The other half of the backwards n-gram model involves the usage of n-grams to fill in the lines preceding the end-of-line words. This was done with the bigram and trigram dictionaries. The n-gram generator took in three arguments for filling in the lines, they were: the number of available syllables left (thirteen allowed syllables minus the number of syllables in the end-of-line word), the already selected end-of line rhyme word, and the one

thousand similar words. Using these three arguments, the n-gram generator was capable of filling in the lines.

The usage of bigrams took precedence over trigrams in the process of generation. But both dictionaries were used to extract potential words for filling in the lines. The process of finding possible words involved the use of the *syllable frequency* dictionary, which tested the similar words and kept those that fell within the constraints of the available syllable slots. From the words that remained, came the elimination of the end-of-line words because they were already in use. The words left were all seen as potentiel fillers.

Once the generator was left with these select options, it needed to gather more words to have a full line equal to thirteen syllables. So, the generator used the number of remaining syllables and selected random words using the values in *phonetic form frequency* dictionary as the weights. This simple process would repeat itself recursively until it had the adequate number of syllables needed to fill in a line. However, through the process of getting more words to fill in the line was a checker to verify that no words were being used twice as one of the filler words. Each time a word was chosen, it was tested as to no overpass the needed syllable amount with its syllable count extracted from the *syllable frequency* dictionary and number of syllables left to fill. The n-gram generator was also apt in testing other words aside from the already selected potentials for the line in between the recursion of selecting words. This helped in avoiding double and having unique words for each line.

To conclude, the n-gram method filled the lines preceding the end-of-line words throughout the generated verse. It implemented different dictionaries and some were used as weights to choose the most optimal words for the line without exceeding the maximum number of syllables.

Creating An Original Verse

Once the line-final words and the n-gram words (preceding these line-final words) were chosen, they were finally formatted into their respective lines to create *one* uniformed rap verse.

Organizing the verse was a small task in comparison to the generation. However, the chosen words for each line were still in their phonetic forms. The lines were transcribed back into their alphabetical form using the two dictionaries. First, their alphabetical forms were gathered using the *phonetic-form-to-word-variants* dictionary. Second, the generator randomly picked one of the variants while implementing weights. The weights for the variants came from the *word frequency* dictionary's values. Finally, the verse was ready to be presented in its alphabetical form as a result of this transformation.

Here is an example of a rap verse with the standard constraints of thirteen syllables required for each line and the *AAAA BBBB C DD* rhyme scheme. This example uses the input words *generation complete*, it is one of ten verses generated. The *word2vec* model for this example was trained on a window size of fifteen. The input theme was the two words *generation complete*.

It and I communication capabilities (A)
FIGHTER house deep the Whodini my communities (A)
Many Buick a sanctified like social Studies (A)
Saturday just like about you dominant species (A)

The address us no head it mist you're with dialect (B)
I thing know for her I'm still a verbal Architect (B)
Yea gotta ass me most babies every aspect (B)
How start it of F-A Fort You shit up the subject (B)

Seventh up a know did what same specific intent (C)

Feel be the them in listening bout club that lyric (D)
Up doubtin my things ever still I through the music (D)

FIGURE 6: Output from the generator with the two terms *generation complete* as the input.

Evaluation

I. Rap Generation Assessment

The following tables present different outputs from the generator using several input theme words and phrases. In each table, there are two syllable outputs. The first column across the four tables contains the standard thirteen syllables while the second column starts with seven syllables in the first table and decreases in syllable length from table to table, ending with a table exhibiting a rap with four syllables to each bar. The columns with less than thirteen syllables were experimental verses. These smaller experimental verses were meant to test whether or not they would be more or less grammatically coherent since less words would be involved.

INPUT	*To the left*	
SYLLABLES	Thirteen syllables per bar	Seven syllables per bar
RHYME SCHEME	AAAA BBBB C DD	AAA BBB CCC DD
Than got of workin rated punishment punishment Your winnin it all and Derty at My Element Andrew the big Keep feel of for That's way a patient Headphones know or said I work we not to this moment BE pointing rock Naked say a word ya line and tied It did me I Bangkok funky the LA gets your pride Time to it Jim money if biggest is You my stride On truth real think the a like knows slut it in wayside My with aftermath Yo sweet milli this in I shot Shit With your on large that rap the got to Chicago A Segundo ain't wait with hoochies and your Elbow		More Bandana that moment Glocks want after gone Distant Has But defense a basement You bush speak know we too fight Two MC All few voice hook right Walk slide new a leave York night New do the now in to Da My But still Fred on out Tha Your head since just me that ta Got ain't dream me everything Know for killers I west wing

TABLE 16: A comparison between a verse of thirteen and seven syllables.

These two verses were generated in different runs yet only share one duplicate for the end-of-line words, which is *moment* in line four of verse one (thirteen syllables) and line one in verse two (seven syllables). In addition, the end words share a strong correlation to the input words. For instance, *stride*, *wayside*, *distant*, *right*, and *wing* share clear links to the input upon personal observation. Moreover, in line one of verse one there is a duplicate with the word *punishment*; proving that potential duplicates are possible in the output lines.

Another interesting point is the difference in rhyme schemes. The two verses have different rhyme schemes, and yet in each one there are perfect rhymes to match the scheme. This is important because it shows how the generator can produce different rhyme patterns. It also shows how the generator can produce more or less complex rhyme schemes.

INPUT	*Driving into the sunset*	
SYLLABLES	Thirteen syllables per bar	Six syllables per bar
RHYME SCHEME	AA BB CC DD E FF	AA BB CC DD E FF
And this what me Buh deep close knew I'm drivin outside That up be diamonds picture the It hood They You're tied Deuces trust we then feelin see of try the benches Shit's stackin a fifth ain't Street there's like Parked garages They Keep payin woman a pleading now the driveway Do be raw Born it California blind The Ave Magazines bitches stay is set got Where The windows Where remember And She time not no bitches this rose It to know disappear me Chevrolet Impalas At how don't my leg Cruz but trust Ridin tit I stroll Independent a ride I'ma I drop yo' petrol		Door a-yo Chicago And D's Oh a metro Decision A curfew Every Avenue Gat up to at sundown I 'Round Westside midtown I'm in Mississippi Somethin to I'm I See Throw 'em to don't dirt bike Ace school and not that pole Be a border patrol

TABLE 17: Two raps of different syllable lengths using the theme, *driving into the sunset.*

These two verses are rich in semantic soundness to the input. They both offer words in the lines and in the end-of-line words that tightly correspond to the input, to illustrate: *driving*, *garages*, *hood* (of a car), *parked*, *windows*, *driveway*, *Chevrolet*, *Impalas*, *Ridin*, *petrol*, *ride*, *metro*, *avenue*, and *sundown*. This direct correlation of input words to the verses proves that fifteen is a good window size for training the *word2vec* model.

On the other hand, the shortening of the verse in terms of syllable length did not vastly improve the quality of the verse. However, some lines are coherent, for instance: *I 'Round Westside midtown*, *I'm in Mississippi*, and *Be a border patrol*. Alternatively, incoherence is present in verse one's lines, but within the lines of this verse there is an interesting use of vocabulary and short phrases, in particular, *They Keep payin'*, *I stroll*, *Where The Windows*, and *Then you're tied*. These implemented phrases, from the generator, aid in understanding the lines more fluidly. They also add to the narrative of the verses.

INPUT	*Baby*	
SYLLABLES	Thirteen syllables per bar	Five syllables per bar
RHYME SCHEME	AAAA BBBB C DD	AA BBB CCC DDD
It rep wigga it's know on when to start the ego Would can't Let I shift the more Tense was when clan Oh Ohhh I times can for okay that's Black cut Miss tomorrow Looking low too of that hollow explode Yo quiero Coat eyesockets and to dipped place show tracks we gotta Me ain't or Maccabee a them you And that nigga Lil I be boy up dun Who's me Chi for just wanna		Dollars as Misses Sho' this And nieces On listen it Clyde Buckin pound too ride Mami friends pride stride Dilla know Captain With can know reason

Ring I things do recall heard Buck to pain so Lisa Girl ain't ya like I'm they keeping none from the Let's ride Bitch a it when Zag me up one Feed my is her knees Switch yay You a uh classic Shady Play light And please	Panties with cousin Know for to gon' By Me and if as PIE Bad this You I cry

TABLE 18: Two raps produced with the theme *Baby* with different syllable lengths.

Both verses share an association with the input *Baby*, for instance: *Yo quiero*, *nieces*, and *cry*. Furthermore, the perfect rhymes in the verses are true to the rhyme scheme and are also true perfect rhymes. Despite the lack of coherence in the lines, the perfect rhymes following the rhymes scheme make up for this because the rhymes add the spirit of rhyming throughout both verses.

INPUT	*California*	
SYLLABLES	Thirteen syllables per bar	Four syllables per bar
RHYME SCHEME	AAAA BBBB C DD	AA BB CC DD E FF
And her rhyme flights game off out I'm care got Boriquas On It's some fright it detail the BACK shottas shottas No in Yeah touchdown And Escobar Murda bizness I'm game lyricist slo-mo Far how bompton Pirus Means girl Tryna through Q Still yaddy it bambi bell Look hamhocks roll Gungan you best my city Cartel Above trust Budget girls he and up damn eyed from Del From An I'm I are who like 'em shiny vein side El J us ahh Dumb we all your past the lame Fresno San Legal it me tell the face ESG Los Angeles Stepped I but you friends We ties P droppin' like Paris	This to Compton Rock is Staten One shanty town Always downtown Lose Catch spike li No run DC Cap My hooooood The hooooooooooddddddddd A scheme the sack Wigs here Chi Chi Choose found to try	

TABLE 19: Two raps generated using the theme *California* across distinct syllable lengths.

The lines produced in these verses share a correlation to the input word *California*, for example, in verse one (thirteen syllables) these terms are used: *Fresno*, and *Los Angeles*. Moreover, in verse two (four syllables) there is also a strong association with the input, in particular, *Compton*, *downtown*, and *hooooood*. The correlation between the input and the verses produced is important because it shows the generator is capable of producing verses that are not only unique but semantically relevant to the input.

Throughout the examples, the generator was able to produce perfect rhymes across the different verses using various input words and phrases. It was even capable of producing perfect rhymes for verses when the syllable length was half the standard thirteen syllables. Additionally, for every verse in tables sixteen through nineteen there is a link between the input and the generated verses. The generator overall achieves the goals in mind for this study and produces original rap lyrics with any given rhyme scheme and length while keeping the input word (or phrases) in mind.

Conclusion

The overall production of rap lyrics across the tables in the previous section (*Evaluation*) is positive. The lines produced are a step in the right direction of automatically generated rap lyrics. The lyrics produced even embody the flow of early hip-hop and rap lyrics subjectively. The following section covers some improvements that would benefit and further the quality of lyrics produced.

I. Improvements

Grammar

The raps produced are more than adequate, however not entirely coherent. For instance, the phrase '*Do be raw Born it California blind The Ave*' (from Table 17) is not grammatically correct. However, a more grammatically correct version could be, '*Do it raw, Born in California, blind The Ave*'.

Adding some form of grammar when gathering new words would greatly improve the quality of lines the rap generator produces. This could be done by implementing simple grammar rules, such as, correct verb usage, or proper usage of prepositions. Finally, implementing a grammar would also help with the flow of rhymes.

A Future with an RNN

A recurrent neural network is a multi-layered artificial neural network. A recurrent neural network can be thought of as a network of loops chained to one another. It is capable of thinking for itself and can vastly improve the quality of the verses produced. In fact, recurrent neural networks are often used for text generation. In theory, it could aid in the coherence of the lines and even more learn to implement more rhymes within the lines since they are only present in the end-of-line words.

In the future, using a recurrent neural network would produce more fluid rap rhymes and lines. It would also be interesting to have the recurrent neural network mimic the style of different artists. This would allow for different styles to be generated. Using a recurrent neural network would unlock endless possibilities and it is the next step for all the work already done in this study.

Bibliography

Caswell, Estelle. “Rapping, Deconstructed: the Best Rhymers of All Time.” *Vox*, Vox, 19 May 2016, www.vox.com/2016/5/19/11701976/rapping-deconstructed-best-rhymers-of-all-time.

Cmusphinx. “Cmusphinx/g2p-seq2seq.” *GitHub*, 12 July 2018, github.com/cmusphinx/g2p-seq2seq.

Connor, Martin. “The Rapper's Flow Encyclopedia.” *Genius*, 28 Feb. 2013, genius.com/posts/1669-The-rapper-s-flow-encyclopedia.

Forman, Murray, and Mark Anthony. Neal. *That's The Joint!: The Hip-Hop Studies Reader*. *Routledge*, 2004, sites.psu.edu/comm292/wp-content/uploads/sites/5180/2014/10/FormanNeal-Thats_the_Joint_The_Hip_Hop_Studies_Readerbook.pdf.

Forman, Murray, and Mark Anthony. Neal. *That’s the Joint!: the Hip-Hop Studies Reader*. Second ed., Routledge, 2012.

“Gangsta Rap Music Genre Overview.” *AllMusic*, www.allmusic.com/subgenre/gangsta-rap-ma0000002611.

“Gensim: Topic Modelling for Humans.” *Radim Řehůřek: Machine Learning Consulting*, radimrehurek.com/gensim/index.html.

Ghazvininejad, Marjan, et al. “Generating Topical Poetry.” Proceeding of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, doi:10.18653/v16-1126.

H. Hinton, J. Martens, I. Sutskever. “Generating Text with Recurrent Neural Networks,” Proceedings ICML, 2011.

H. Hirjee and D. G. Brown. “Automatic Detection of Internal and Imperfect Rhymes in Rap Lyrics,” Proceedings ISMIR 2009, 2009.

H. Hirjee and D. G. Brown. “Rhyme Analyzer: An Analysis Tool for Rap Lyrics”, Proceedings ISMIR 2010, 2010.

Kao, Justine, and Dan Jurafsky. *A Computational Analysis of Style, Affect, and Imagery in Contemporary Poetry*, 2012, www.aclweb.org/anthology/W12-2502.

Malmi, Eric, et al. “DopeLearning.” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*, 9 June 2016, doi:10.1145/2939672.2939679.

Manning, Christopher D., et al. “Boolean Retrieval.” *Introduction to Information Retrieval*, Cambridge University Press, 2017, nlp.stanford.edu/IR-book/pdf/01bool.pdf.

Nguyen, Hieu, and Brian Sa. *Rap Lyric Generator*. Stanford CS224N Final Projects 2008-9, 2009, nlp.stanford.edu/courses/cs224n/2009/fp/5.pdf.

Potash, Peter, et al. “GhostWriter: Using an LSTM for Automatic Rap Lyric Generation.” *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, doi:10.18653/v1/d15-1221.

Shmyrev, Nickolay V. “Sequence-to-Sequence G2P Toolkit.” *GitHub*, github.com/cmusphinx/g2p-seq2seq/blob/master/README.md.

Smitherman, Geneva. *Talkin and Testifying: the Language of Black America*. Houghton Mifflin, 1977.

Wu, D & Addanki, K & Saers, M & Beloucif, M. Learning to Freestyle: Hip Hop Challenge-Response Induction via Transduction Rule Segmentation. 102-112, 2013.