Automatically generating song lyrics based on artist style and theme

Emily Ling
eling8@stanford.edu
Owen Wang
ojwang@stanford.edu
Charissa Plattner
chariss@stanford.edu

1 Introduction

We often hear the phrase, "Imitation is the sincerest form of flattery". And this is certainly true– when we discover something that we like, we adapt and transform it for ourselves. This desire to imitate others is what has inspired many people to work on artificial intelligence projects of imitation. Some people use neural networks to mimic the artwork of others (Gatys and Ecker, 2015); others use Markov models and recurrent neural networks to mimic the writing style of famous authors.

This team decided to tackle the project of lyrical imitation. We met last year in Testimony A Cappella and it was our shared love of music and computer science that first led us in the direction of using AI for something involving music. After considering generating pop musical chord progressions, predicting whether a song would become a hit (Ni and Rodríguez, 2011), and determining which artist a song belonged to, we settled on generating song lyrics based on an artist and a common theme for that artist.

We had to consider two major aspects of the project for our task. The first was figuring out how to cluster songs in such a way that cohesive themes would arise. The second was using AI techniques to generate lyrics in the style of the artist. We will discuss the evaluation metrics in detail later in the paper.

1.1 Dataset

Our dataset is a huge set of musical lyrics matched to artists. We scraped the lyric text from azlyrics.com for approximately 120 of the most popular artists for a total of over 6000 songs. We decided to limit the number of artists to those with a certain threshold of songs, so that we could have enough data for each artist to generate themes properly and create music in their style.

1.2 Input and Output

Our system has a user-interface that allows the user to enter an artist name, check out the clusters developed from that artist's music, and then choose a cluster. The backend takes in that artist name and cluster and outputs the lyrics to a song generated in the style of the artist on that theme.

2 Relevant Work

Because lyric imitation is a fascinating problem, there have been many attempts to tackle it. The first avenue we researched was rap music generation. The article "Markov Constraints for Generating Lyrics with Style," published in ECAI 2012, uses markov chains to generate lyrics with proper style and syntax, while employing constraint satisfaction to ensure rhyme and meter are maintained throughout the lyrics. We actually took a very similar approach; however, our constraints were much looser, as pop lyrics do not have as many constraints for rap does.

An application called Lyricize uses Flask to generate lyrics with Markov chains. This project was not nearly as complex as the previous research link. Lyricize appears to only use n-gram Markov chain generation to get new lyrics. Though we also used simple Markov chains, we also implemented a function we called epsilon random word generation. There was some probability, epsilon, that the next word generated in the line was pulled from an out-of-domain source. We wanted to implement this because an artist does not only use words they have used before in order to write new music; they have to supplement with other words.

Regarding extracting the most relevant words in a theme, we read the paper by Professor Daniel Jurafsky, "Narrative framing of consumer sentiment in online restaurant reviews". However, that team used a technique called "log odds ratio informative Dirichlet prior" to identify the statistically over representative words in various categories. Though we used a much more primitive method, preprocessed weighted counts basically, we produced decent results. However, we have considered that one way to extend our project would be to implement Jurafsky's statistical method.

3 Baseline / Oracle

As a baseline, we are returning a sequence of words where each word is randomly sampled from the corpus of all the artist's words used in their lyrics, weighted with their relative frequencies in the corpus. This baseline also does not make any attempt to imitate song structure, such as constructing verses, choruses, and so on. Each word is generated independently of the others. The idea is that this stream of words will be crudely representative of what the artist sings in his/her music.

An example of the baseline output is shown below. The artist we trained our baseline on is Taylor Swift:

"and up you on worrying always right sound And with in are make-up time in when Just all your drive about And the Oh I've the Or 'Cause New You turned blood mind were I a now Should is scar face that I bitter Everything he's know my fiction to be it this treacherous fiction reason I'm Jesus echoed thought bad It's And I you're I wonderland screaming your musical so the and wish people I'm she's friction 30 about not the at just stranded to skipped The that once did rain? Stephen cookies story pouring just tragic I forget all stay I you York Revolutionaries don't I smiled song what oh lose and eye getting respects"

The oracle for our project is a set of song lyrics written by a human the artist, ideally to emulate the artist's style. A human, assuming adequate knowledge of popular music, will quickly and accurately create a good set of lyrics. A human oracle has advantages over a machine-generated set of lyrics in several ways: his/her grasp of English grammar and syntax will be much better than a computer model, can better assess contextual information in complex ways around the lyrics to fit themes better, and the human can allow for linguistic exceptions to enhance the lyrics' aesthetic. For example, some artists will repeat phrases or lines or use them in a non-conventional manner to increase rhetorical power. Train's line "Reminds me that there's a time to change, hey, hey, hey", in Drops of Jupiter is not a complete sentence, yet it works fine in context of the verse it is in.

4 Model / Algorithms

4.1 K-Means Clustering

When we generate an artist object, we map each song to three theme clusters using k-means clustering. We run k-means clustering over all of the distributed song representations, which use significantly preprocessed bag-of-words features. To do this, we must assume the principle of compositionality, or the idea that we can glean information about the theme of a song by examining its individual words. We are assuming that songs with similar word distributions will also have similar themes.

Our final model extracts nouns, verbs, adjectives and adverbs from each song and then uses the NLTK Snowball stemmer to stem them. Then we get counts of each of these stemmed words. These counts are transformed into a vector representation of that song. From these vectors, we create a song-word count matrix, where each row represents a song in the artist database and each column represents the count of a specific stemmed word. The matrix is first normalized using TF-IDF with add-one smoothing and then dimensionality reduced using Truncated SVD. Then we used k-means clustering with three clusters on those song representations to cluster the songs into groups of similar theme.

4.2 Most Common Words Generator

After clustering songs into three theme clusters using k-means clustering, we wanted to find the best words to represent each theme. We did this by choosing ten representative words that appear most frequently in songs of a particular theme. For each cluster, we iterate through the unigrams, bigrams, and trigrams of each song in that cluster and sum together the total number of times a particular n-gram appears in a cluster. We hypothesized that the words most strongly associated with a particular theme would appear most frequently in that theme, so we took the ten most common unigrams from each theme. To avoid trivial words such as pronouns, be verbs, and nonsensical words, we used the NLTK stopwords corpus in English to filter out common words and the PyEnchant English dictionary to ensure that all representative words and real English words. Additionally, we used the NLTK part of speech tagger to only consider nouns, verbs, adjectives, and adverbs, in order to avoid picking words such as conjunctions and pronouns that are less meaningful. These filters allowed us to pick more substantive representative words for each theme.

4.3 Lyric Generation

Before lyric generation begins, we first learn all the information from a single artist and his/her songs. The counts of each unigram, bigram, and trigram of words that appeared in the artist's songs are stored. Each line is analyzed without wraparound. In addition, we perform theme clustering on the artist's songs as described earlier, and create a dictionary of weights themevalues that register each N-gram's degree of association with the three themes. First, each of the artist's songs is assigned to a cluster. Then, for every uni/bi/trigram that appears in any song, we store the number of times that N-gram appears in themes 1, 2, and 3 in theme-values under the key of that N-gram. This information is used later in line generation.

Our lyric generation model is primarily based on a N-gram model that incorporates unigrams, bigrams, trigrams, and theme association data to generate lines of lyrics. To limit our project to a manageable scope, we decided to generate songs with a hardcoded structure. Each song would

consist of six song blocks: first verse, chorus, second verse, chorus, bridge, chorus. After the user specified an artist and theme, the process would begin.

Each block would be composed of somewhere between 8-12 lines, the exact number being uniformly distributed. Each line is generated based on the given artist and theme; this is described in more detail in the next paragraph. Block generation has a few caveats. At a small chance (10%), an odd-numbered line will repeat the line before it, but only if the most recent odd-numbered line before it did not repeat. We implemented this feature to imitate how artists' lyrics tend to repeat themselves and are not constantly changing from line to line, but also did not want repeats back to back as it would seem excessive.

To generate each line, we would first begin by selecting a trigram in the artist's corpus with weightedRandomChoice(). Then, there are many possibilities for the next word to pick in the line. We score each possible next word W, following the two previous words first and second with the following formula, then select one based on weightedRandomChoice():

Let unigram = W, bigram = (second, W), trigram = (first, second, W).

Let freq(X) := frequency of the N-gram X in the artist's corpus.

Let theme(X) := frequency of the N-gram X in songs from the artist of the chosen theme.

$$score = 100 * log(freq(trigram) * theme(trigram) + 1)$$

$$+10 * log(freq(bigram) * theme(bigram) + 1) + 1 * log(freq(unigram) * theme(unigram) + 1)$$

Words with a trigram score (the product of freq() and theme()) of zero, meaning they did not ever follow the two previous words in the artist's corpus, were excluded from consideration when picking a next word. Our hypothesis was that including these words will decrease the quality of our generated lyrics because they are unlikely to fit together well with the previous two.

The largest problem with the algorithm described above is the limited vocabulary our program has. By picking words that are already existing in the artist corpus, we necessarily limit the authenticity of generated lyrics by limiting data. For example, often our algorithm would run out of trigrams to use and end a line prematurely. To mitigate this problem we drew inspiration from the epsilon-greedy concept introduced when we covered learning and games. At a small chance epsilon (20%), the lyrics generator would pick one word from the unified corpi of all the artists' songs we have, still using the same algorithm. Using a broader corpus has the tradeoff of increasing our domain of words to choose from, but decreases lyric generation's accuracy.

5 Results

5.1 K-Means Clustering

The following are the B3 F-1 scores of the three song clustering algorithms we implemented.

A Comparative Look at Clustering Algorithms

Artist	**Taylor Swift**	**Ariana Grande**	**Owl City**
Random-Baseline B3 F-1	0.3383	0.3922	0.3464
Bag-Of-Words B3 F-1	0.4289	0.4825	0.3828
Stemmed-Words B3 F-1	0.4507	0.5260	0.4827

5.2 Most Common Words Generator

The following is an example of the representative theme words generated for Train, after clustering songs by Train and picking the ten most common words in each cluster, removing stopwords, non-English words, and words that are not nouns, adjective, verbs, or adverbs.

Theme 1: always, open, goodbye, way, ground, jump, call, burn, know, hey
Theme 2: bring, baby, snow, roll, bright, shake, merry, wake, happiness, ho
Theme 3: everything, could, never, everybody, oh, eh, back, get, got, love

5.3 Lyric Generation

Below is a song generated for Owl City based on a theme consisting of the following most-relevant words: [always, gold, fire, world, top, good, way, time, shine, tonight].

[Verse 1]
even though i'll never be among
looks a bit surreal
my burns were third-degree
a ghost of a smile of clean pearly whites
and no one recognized me i suppose i would just stand and
i heard a gorgeous sound
these gloomy nights wear on
will ever know
i searched again and again

[Chorus]
blankets on your ceiling
a bomb she's an atom bomb
turn light blue
to the willow trees are waving 'til we come back
when i sang i'd rather drive whisperings
on your holiday
and if my my
their eyes out

[Verse 2]
and all i pay is my drive
when i'm in if you're the only north
crash and burn where fire and ice collide
there and it's crystal clear i can taste your lips
not believe your eyes whispered disguise
if my echos can reach your ears
angel eyes beautiful exactly
my window somewhere way up in your heart pound
bursting at the hip yeah your sidekick needs you

[Bridge]
sing dream like i've never been so afraid
stay on track so i watch you sailing far sugar
to prove the world fly by feather this try be
the bed rock

6 Discussion / Error Analysis

For evaluation, we had people cluster the songs of some artists into three clusters according to theme. We then used B3 and pairwise precision, recall and f-1 scores to see how our various models performed against these human-annotated clusters.

As a baseline for clustering, we randomly assigned each song to a cluster. These are the results we saw from that:

This random model performed as we expected. Because there were three clusters, we would expect these values to hover around $\frac{1}{3}$ as well.

We then decided to use k-means clustering to cluster the songs. We created a song-word count matrix, where each row held the vector representation of the song. There was no preprocessing of the words. After creating the matrix, we ran TF-IDF to help normalize and smooth and then used

Random-Baseline Clustering Evaluation

Artist	B3 / Pairwise Precision	B3 / Pairwise Recall	B3 / Pairwise F-1
Taylor Swift	0.3388 / 0.3272	0.3378 / 0.3262	0.3383 / 0.3267
Ariana Grande	0.4605 / 0.4332	0.3416 / 0.3144	0.3922 / 0.3644
Owl City	0.3436 / 0.3247	0.3492 / 0.3296	0.3464 / 0.3271

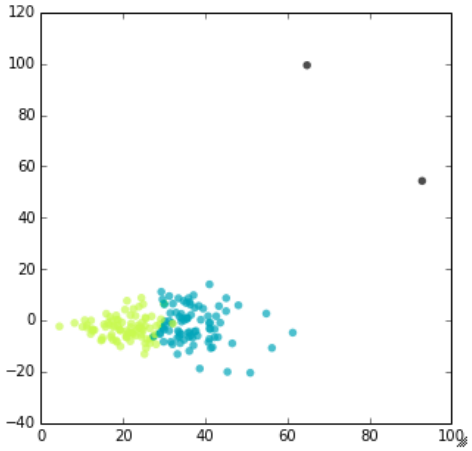
Truncated SVD to reduce the dimensionality of the matrix. Then we ran k-means clustering of these song vector representations.

Bag-Of-Words Clustering Evaluation

Artist	B3 / Pairwise Precision	B3 / Pairwise Recall	B3 / Pairwise F-1
Taylor Swift	0.3558 / 0.3446	0.5398 / 0.5345	0.4289 / 0.4190
Ariana Grande	0.4653 / 0.4353	0.5011 / 0.4704	0.4825 / 0.4522
Owl City	0.3640 / 0.3422	0.4037 / 0.3859	0.3828 / 0.3627

We saw there was significant improvement with recall, but almost no improvement in precision. After examining a graph of the clusters, we saw that almost all the songs were very tightly clustered in one region of the graph. There was very little spread of the vector representations, which we had squashed down to 2D.

Figure 1: Taylor Swift song clustering with first attempt at k-means: We see here that the truncated 2D vectors are very densely packed in one corner. It seems like the hundreds of vector dimensions are not helping the clustering algorithm, because the clustered vectors are near each other even after being squashed down to 2D space.



We realized that we should only be taking into account nouns, verbs and adjectives, because there are more important in clustering themes than pronouns, conjunctions, etc.

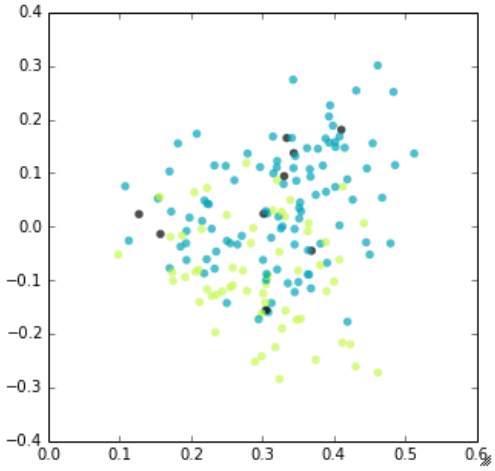
For our final model, we decided to use NLTK to filter out words that were not nouns, verbs, adjectives or adverbs. We then used a Snowball stemmer to get the stem of each word. We realized that words like ‘loving’ and ‘love’ should be in similar themes, but they were separate features in our current model. We created a song-word matrix again, but with the filtered, stemmed words. We also used TF-IDF and SVD to collapse the matrix.

Stemmed-Words Clustering Evaluation

Artist	B3 / Pairwise Precision	B3 / Pairwise Recall	B3 / Pairwise F-1
Taylor Swift	0.4469 / 0.4306	0.4546 / 0.4455	0.4507 / 0.4379
Ariana Grande	0.4961 / 0.4756	0.5597 / 0.5044	0.5260 / 0.4896
Owl City	0.3821 / 0.3484	0.6553 / 0.6477	0.4827 / 0.4531

This raised our precision significantly! We also saw that the 2D vector presentations for each song were much more scattered, so the clusterer was able to find more nuanced clusters.

Figure 2: Taylor Swift song clustering with final k-means: We see here that the clustering algorithm is finding clusters that aren't necessarily near each other in 2D space, meaning the many dimensions are contributing.



To find the most common words, our first attempt was to pick 10 representative words in each cluster. For each word, we calculated the percentage of its occurrences in each theme. Then for each theme, we picked the ten words that had the highest percentages for that theme.

However, we realized the words that were just very common ended up as the representative words. For example, our first attempt for Ed Sheeran Theme 2 produced: "tell, make, i'll, rain, it's, you're, her, need, don't, i'm". So we decided to make sure that the words are nouns, adjectives or verbs. Then we got "rain, hallelujah, live, make, it's, i'll, you're, need, don't, i'm " for Ed Sheeran Theme 2.

Next, we decided to remove contractions and the stopwords in the NLTK stopword dictionary and got "keep, life, eyes, take, go, know, say, get, let, never". Lastly, we decided to enforce that the words be valid English words.

Our final model for Ed Sheeran Theme 2 produced the representative words: "makes, man, waving, city, seem, still, tell, make, rain, need". There is still some work to be done, but this is a significant improvement from the first attempt.

To evaluate the quality of our song generation, we created an evaluation rubric to numerically score actual songs and generated songs based on several different factors. The two broad categories for evaluation were: 1. Artist and theme fidelity, and 2. Song fluency. The first category helps us to evaluate how successful we were at imitating a particular artist, and how clearly we were able to identify and maintain distinct themes. This category was broken up into theme fidelity and artist fidelity, each of which was given a score between 1 and 5. The second category, song fluency,

focuses on how well our generated song mimics an actual song. This category contained metrics for line fluency, overall cohesion, and song structure, each of which was evaluated on a scale of 1 to 5.

Our evaluation rubric also contained a question that would serve as a kind of Turing test. This question asks whether a given song seems like a real song or a generated song. In an ideal situation, generated songs would be able to pass as real songs by the artist, and if we are able to pass this test, this would indicate that the fluency, cohesion, and style of our generated songs are on par with actual songs written by the artist. The full evaluation rubric is in the appendix.

To evaluate the songs, we created a dev set by pulling three random songs from five popular artists (Owl City, Taylor Swift, Ariana Grande, Adele, Jason Mraz), which would serve as a baseline measure for how well actual songs would perform using our evaluation rubric. We normalized these songs to have the same syntax and capitalization as our generated songs. Then, we generated three songs for each of those five artists, for a total of six songs per artist, half of which were generated. Next, we surveyed several people, who each read a set of song lyrics and filled out the evaluation form. A total of 34 evaluation forms were completed. The results are below.

Statistics for Evaluation Scores for Actual vs. Generated Songs

	Artist Fidelity	**Theme Fidelity**	**Line Fluency**	**Overall Cohesion**	**Song Structure**
Actual songs					
Mean	3.00	4.10	3.86	3.90	4.24
Median	3	4	4	4	4
Standard Deviation	.894	0.625	0.793	0.831	0.831
Generated songs					
Mean	2.64	3.18	2.27	2.36	3.36
Median	3	3	2	3	4
Standard Deviation	0.809	0.874	0.786	1.206	0.924

Although the scores for generated songs were clearly lower than scores for actual songs, they were still somewhat comparable. The medians for artist fidelity and song structure were identical for both actual songs and generated songs, indicating that our algorithm did fairly well at matching artist style and producing lyrics that were in the structure of a song. Actual songs performed consistently better in the category of theme fidelity, indicating that our generated songs did not match the theme as well as actual songs that were clustered into a theme. The largest gaps in performance were in the categories of line fluency and overall cohesion, which are the two areas in which our algorithm could use the most improvement. These results indicate that future work should focus largely on these two categories to make generated songs more grammatically correct and more focused and cohesive throughout.

Additionally, we also collected information about whether people believed each set of lyrics represented a generated song or an actual song. The results are below:

The results for our generated songs indicated that we failed to pass our Turing test, meaning that our generated songs were not good enough to truly pass as actual songs from an artist. However, a surprising result was that a significant portion people thought that real songs were actually generated, indicating that some real songs are somewhat nonsensical and do not fit into the typical

Percentage of Votes for Actual vs. Generated Songs

	Percentage of Votes: The song was real	Percentage of Votes: The song was generated
Actual songs	80.95%	19.05%
Generated songs	0%	100%

format or style expected of a song. Although this percentage was somewhat low, it indicates that perhaps the standards or expectations people hold about lyrics are rather high, which makes the task of making generated songs seem like actual songs more difficult.

7 Conclusion

Although our algorithm was not able to generate lyrics that appeared equal to real lyrics, we were able to achieve a significant improvement to our baseline algorithm and were able to effectively implement a Markov model and K-means clustering to generate lyrics that were fairly realistic. However, our algorithm is imperfect and can be improved in many ways. One interesting thing that we noted was that our clustering algorithm did an excellent job at isolating songs that are covers, or songs that belongs to another artist. For example, the clustering for the artist Hozier put a song called 'Problem' alone in one cluster. Unsurprisingly, this song was actually originally done by Ariana Grande, an artist whose style is diametrically opposed to Hozier's. We hypothesize that removing these covers from the data would actually help our clustering algorithm a lot. For a future project, we are considering cleaning up our data significantly to remove these covers, so to avoid noise in our results.

Going forward, there are many additional features that could be implemented to improve our algorithm. As stated in the results, two major areas of improvement are line fluency and overall cohesion. To improve line fluency, we would need some way to enforce grammar on each line, so that grammatically impossible phrases are removed. We could use libraries such as pylinkgrammar after generating each line to ensure that the lines are more fluent. Additionally, we could require that the first word of each line is actually a word that begins a line in a real song. For example, it's highly unlikely for a song lyric to start with "me" and still be a cohesive sentence. Some words are much more likely to be used to start a line than others, and by doing a weighted random choice over all words that an artist uses to begin lines, our lyrics would likely become more fluent.

To address overall cohesion, we would need to find ways to make our generated songs adhere more closely to a specific themes. This would involve playing with our k-means clustering algorithm to find ways to cluster songs more accurately, and perhaps improving the way that we find the overrepresented words in each theme. Our current algorithm simply picks the most common words in each cluster that are "significant" words (eliminating stopwords and only considering certain parts of speech), but there are many better algorithms for doing this. One such algorithm is Log Odds Ratio Informative Dirichlet Prior, but many such algorithms exist and further research into this area would be helpful. An improved method of picking words from a theme would also help us to better adhere to the theme in generated lyrics. Another idea would be to expand our algorithm to include features that take into account line wrap-around, in that adjacent lines should have much more similar words and themes than non-adjacent lines.

Additionally, songs typically have structural characteristics such as variations in verse and chorus length and order, rhyming words at the end of lines, and repeated phrases or lines. These characteristics are essential in making a song more realistic, so for future work, one possible idea would be to learn song structures based on stanza length and repeated sections, and use that information to generate likely song structures for each artist to be used for generating songs. Additionally, it would also be possible to use a CSP model to enforce characteristics such as rhyming, song structure, syllable count, and syllable stresses. This would likely help the song to seem more cohesive and structurally sound, as well. Although there is significant work left to be done before this algorithm can generate lyrics that appear to be real songs, we have greatly enjoyed exploring this topic and hope to continue this work in the future.

Acknowledgments

We would like to thank Arzav Jain for helping us narrow down the scope of our assignment and for providing excellent advice. We would also like to thank the CS221 course staff for writing the excellent weightedRandomChoice() function from the cars assignment that we used for our Markov model.

References

[1] Gabriele Barbieri et al. "Markov Constraints for Generating Lyrics with Style". In: *ECAI* (2012). URL: https://www.csl.sony.fr/downloads/papers/2012/barbieri-12a.pdf.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *CoRR* abs/1508.06576 (2015). URL: http://arxiv.org/abs/1508.06576.

[3] Dan Jurafsky et al. "Narrative framing of consumer sentiment in online restaurant reviews". In: *First Monday* 19.4 (2014). ISSN: 13960466. URL: http://firstmonday.org/ojs/index.php/fm/article/view/4944.

[4] Real Python. *Lyricize: A Flask App to Create Lyrics Using Markov Chains*. 2014. URL: https://realpython.com/blog/python/lyricize-a-flask-app-to-create-lyrics-using-markov-chains/.

[5] Shlomo Dubnov and Gerard Assayag and Olivier Lartillot and Gill Bejerano. "Using Machine-Learning Methods for Musical Style Modeling". In: *IEEE Computer Society* (2003), pp. 3–10. URL: http://musicweb.ucsd.edu/~sdubnov/Papers/CM.pdf.

[6] Yizhao Ni and Raul Santos-Rodriguez and Matt Mcvicar1 and Tijl De Bie. "Hit Song Science Once Again a Science?" In: (2015). URL: http://www.tijldebie.net/system/files/MML2011-final.pdf.

Appendix

7.1 Clustering

The following are cluster graphs of songs by two artists, Hozier and Ariana Grande.

Figure 4: Hozier, Version 2

Figure 6: Ariana Grande, Version 2

7.2 Lyric Examples

7.2.1 Initialize Taylor Swift line generation

1. that's how high tide came up above you
2. looking at the conversation with shaky hands of kissing in her bedroom"
3. you're somewhere where else to the milky way before fire jumping through that song
4. you're staring at things that can't stop won't tell

7.2.2 Ariana Grande song generation without random epsilon insertion and theme fidelity

This was generated in the style of Ariana Grande without any themes.

[Verse 1]
i really need you your love is all i really want you now
you think you know i shouldn't ever call back
a little drunk
baby i been livin' my life is easy
talkin' bout movies don't need a reason to
i pick myself up i pick myself up i
i still can't handle all the drama boy it's overkill
got your pride
my heart is stuck in traffic

[Chorus]
can't let it snow in california
got it everyday

and i break free
’cause every time my boyfriend and i lose all track of time
lovin’ it baby x3
we start turning around
i get a little bit breathless
i think i love it i got i got that jam make them lose
all i can take it higher

[Verse 2]
tu me prives
me feeling emotions
we’ve been friends so long
i been feeling you
so what we had
take my love but it all but you
to me i can hold on hold on hold on to
let me see you rock to the door of your charm
all away if he won’t be cause it

[Bridge]
with the hand that
i’m a hot car ride
of it’s written on your body
i don’t even get halfway through oooh

[Chorus]
can’t let it snow in california
got it everyday
and i break free
’cause every time my boyfriend and i lose all track of time
lovin’ it baby x3
we start turning around
i get a little bit breathless
i think i love it i got i got that jam make them lose
all i can take it higher

7.2.3 Ariana Grande Christmas theme

This was generated in the style of Ariana Grande, using the Christmas theme.

[Verse 1]
would come true
you don’t you don’t like it like this
give a wink give a little time look
herc was on the dance floor i’m michael jackson
fall in love again if he won’t be
let it snow in california
by the fireplace oh-oh-oh

’cause i can’t explain what i’m feeling boy

[Chorus]
let it snow let it snow is blasting
get in the air
give it all but you turn right over to the butterflies
’cause i can’t wait for the impossible
let it snow is blasting out
fall in love again if he won’t be christmas it

[Verse 2]
i’m asking for the holidays
give it all in me
give it all away if he won’t be cause
let it snow let it snow let it snow in
fall in love again if he won’t be it won’t
tell me if you like it when you touch me tenderly
in love again if he won’t be back again
tell me if he really cares

[Bridge]
let it snow in california
let it snow in california
let it snow let it snow let it snow let it snow let
give it all you ever need to be my baby right here in

[Chorus]
let it snow let it snow is blasting
get in the air
give it all but you turn right over to the butterflies
’cause i can’t wait for the impossible
let it snow is blasting out
fall in love again if he won’t be christmas it

7.2.4 Chorus generation

This chorus was generated for Zac Brown Band.

[Chorus]
the sweetest winds they blow across the room with a knife
see the problem is that you’re right there
in the world away for a queen
heart still beats fast for you climbs
and every lesson i have lived in a cadillac
bone on the way i would trade a thousand babylons
and i cherished every time i turn that truck
back and smell the sun upon my skin
on a permanent vacation

but not least
don't need those
in the ocean is my only medication

7.3 Generated Lyrics Evaluation Rubric

[Theme and Artist Fidelity]

Theme Fidelity: How well does the song maintain the theme? Are the words in the song related somehow to the theme?
1: Does not match theme at all, 5: Matches them perfectly

Artist Fidelity: Could this be a song written by the artist? Are there phrases that you could hear the artist using?
1: Does not match the artist, 5: Matches artist perfectly

[Song Fluency]

Line Fluency: Do individual lines flow with respect to English grammar rules?
1: No line fluency, 5: Grammatically perfect

Overall Cohesion: Do the verses and choruses go well together? Do the verses make sense with each other?
1: No cohesion, 5: Fits together extremely well

Song Structure: Do the lines rhyme? Is the bridge shorter than the verse and chorus? Do the lyrics form the structure of a song?
1: Does not seem like a song, 5: Clearly represents a song

[Turing Test]

Do you think this is an actual or generated song?
- Actual song
- Generated song