Machine Learning Engineer Nanodegree - Capstone Proposal

Music Information Retrieval and Emotion Annotation using Lyrics

March 11, 2018

Huzefa Raja

Domain Background

Music Information Retrieval[1] is performed in many ways, and usually involves musical feature extraction from the audio. Features such as the **Mel-Frequency Cepstral Coefficient (MFCC)**[2], the key, chords, harmonies, melody, main pitch, beats per minute or rhythm in the piece, etc. are extracted. Information about the audio helps with **Acoustic Fingerprinting**[3] (which can be used for identification of music), **Score Following**[4], etc. And **music recommendation**[5] works in various ways - such as keeping a track of users' listening history and using metadata in combination with it to suggest music.

Another important aspect of music (unless it is instrumental) are the *lyrics*. In this project, I wish to perform MIR using lyrics, and then with the help of the extracted features, I wish to **annotate emotions to songs** that are sent in as input to a deep neural net. For the MIR part, I wish to use **Topic Modeling**[6], and for annotation, I wish to perform **Multi-label Classification**[7].

Personal Motivation

This is a bit anecdotal, I have always been a fan of music and have wanted to create a project that involves music in some way, and is not just an android music player (which is something I work on on and off). There are two songs in particular that I would like to point out to make my point: **Nothing by The Script**[8] and **Marvin's Room by Drake**[9]. The metadata for these songs does not have much, if anything at all, in common. While an

argument could be made for either of them having elements of R&B, because of the artists and the albums they are present on, the genres do not match (the albums assign them Pop & Hip-Hop/Rap respectively). The artists also haven't collaborated for any sort of link to be created by a recommender system unless a link is formed using users' listening habits.

However, if one were to **understand the lyrics**, they would know that *both of those songs involve the protagonist (singer) drunk dialing a former significant other*. This similarity is not evident by the metadata or the audio itself, but via the lyrics. Since I realized this I have been wanting to make a system that identifies such relationships in music.

I would like to mention here that I do not believe that I as of yet possess the skill-set to perform such an operation, in this project I am attempting to perform a lite-version of that. The two songs above, for example, if tagged with the topic of (say) drunk-dialing, may be assigned **[love, sorrow, desolation]** (for example) as the emotions. Someone looking for music that represents or resonates with a particular emotion (for instance anger or guilt, though some may argue the latter is a feeling, but let's not get into that) may find such annotation useful (using perhaps an app that uses a dataset output by this project to query for songs).

Problem Statement

The problem is twofold:

- ❏ The first part of the problem involves **preparing the dataset for classification** using **clustering**. The input dataset contains information such as title, artist, year, genre, and lyrics, the last of which should be used as corpus by a Topic Model to cluster these songs.
- ❏ Each cluster should have **a set of emotions associated** with it, which should then be used to perform multi-label classification on the dataset.

Simply put: given lyrics, the end goal of the project is to annotate it with the emotions that are prevalent in the song.

Datasets and Inputs

The dataset for the project can be obtained from Kaggle, at this link.

The dataset contains the usual metadata about songs, including lyrics, which would require cleaning (elaborated in Project Design). It has lyrics for **380,000+ songs** (about 266000+ have non-NaN lyrics), so there may be enough data for this project. Depending on time constraints and performance of the machine used for training & testing and the depth & complexity of the network, it may not be necessary to use the entire dataset. Also, songs with only English lyrics would be used.

After the first part of the problem is solved, the dataset would be augmented with a few additional columns (equal to the total number of emotions).

Assigning Emotions

The original idea did not involve a Topic Model. It was to find a dataset that contained songs assigned with emotions, and combine it with another that contained lyrics (or finding one that contained both). While I found a dataset of lyrics, I did not come across any for emotions; the ones that I did find (and was not necessarily able to access) were either annotated with genres (and not emotions) or with the annotations being based on the sound itself rather than the lyrics, or had a simplified set of emotions akin to sentiment analysis (happy, sad, or neutral). I also came across this amazing piece of work, where NLP is used to perceive many of the facets of music via lyrics, but not necessarily the emotions. I learned about what Topic Models are en route to abandoning the brave idea of manually assigning emotions to the dataset. While that is possible, it should probably be done by experts in music and/or linguistics.

Unfortunately for me, I have neither the expertise nor access to such people, nor do I possibly have the time. So I decided to make do with the following:

- I would need to start finalizing the list of possible emotions for the project. To do so, I would refer to Robert Plutchik's Wheel of Emotions model. Emotion theory is actually quite interesting and I would look into it as well during the course of this project to get insights. This task would be difficult to complete without a topic model ready, so this "finalization" would be an ongoing process.
- I would then use a Topic Model to create a sensible division in the dataset (in other words, find the number of topics via experimentation). Given the nature of the problem (and the nature of the text), there may be multiple solutions and non-solutions. Within all that, I would have to find something sensible.
- Next, manually assign emotions to the topics. This would be difficult. I've gone into detail about this a little further ahead. The gist of it is: assuming the total number of emotions is 10, and the number of topics is 30, I would manually assign a set of emotions to each of the topics.
- Each song would thus be assigned the emotions of the *topics* that are the most prevalent in its lyrics. *Topics* here is plural, and I do hark back to this point later in the document.

This approach *does* have issues:

- Highly subjective (and I am no expert)
- It may still group some dissimilar songs
- Less pervasive topics are ignored
- In spite of what I say in the paragraph below, it still is very difficult.

It is possible that it would save quite some time (assigning emotions to **x topics** would be much faster than assigning them to **y songs**, where x is possibly in *tens* while y is in *thousands*).

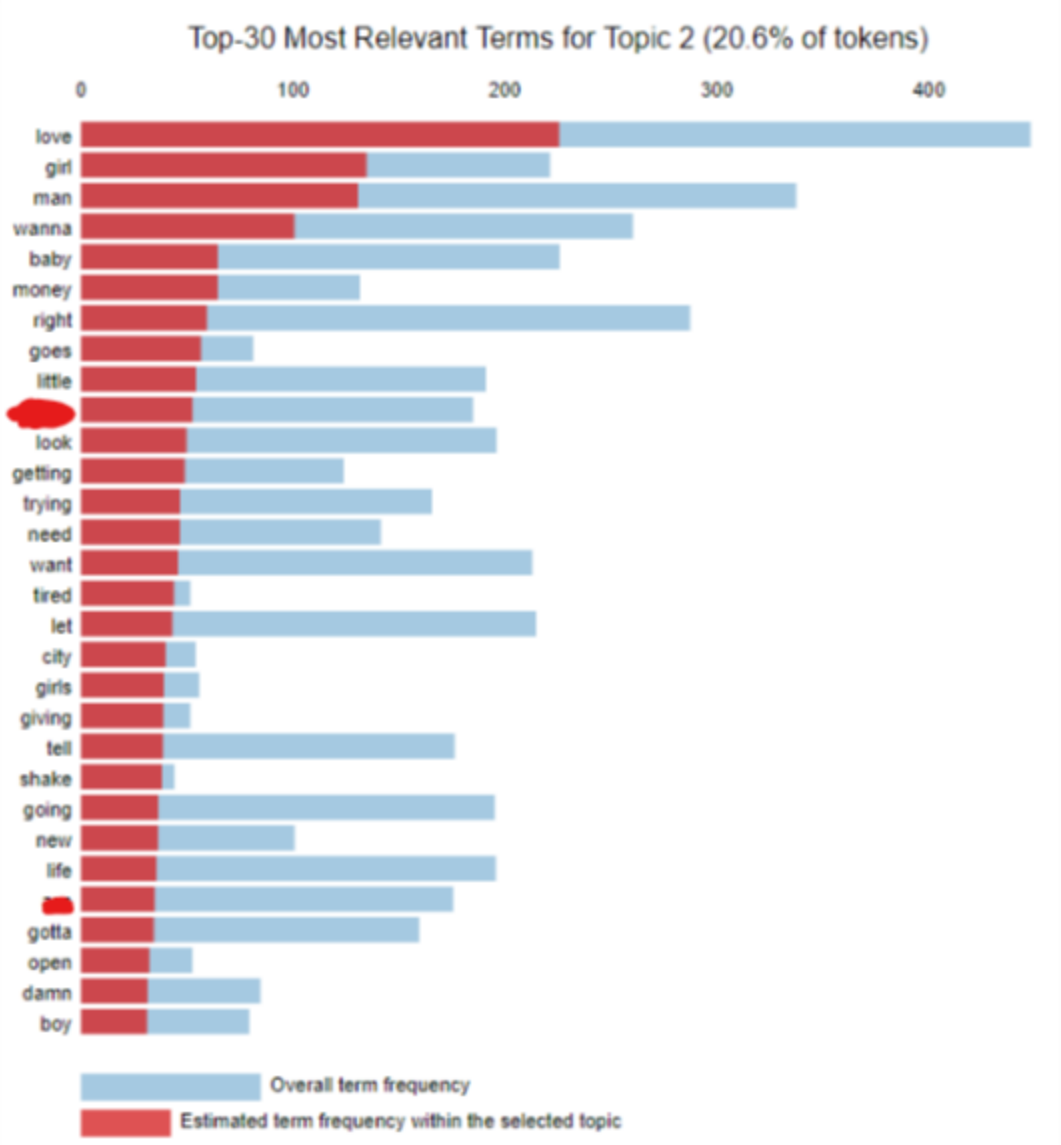
The final dataset for classification would have lyrics and emotions.

Assigning Emotions, an example:

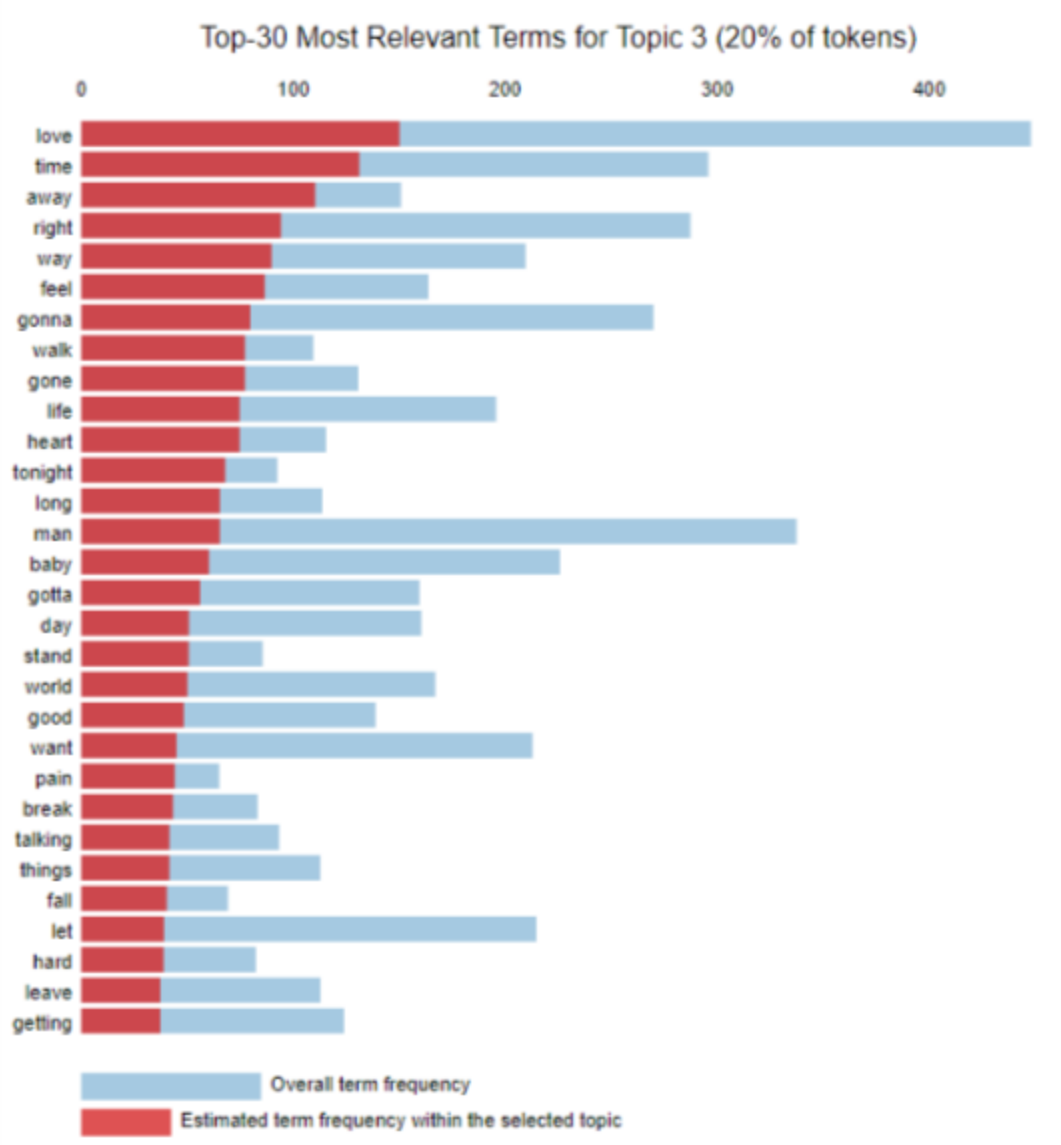
The following visualizations were created using pyLDAvis. The model visualized here is **Latent Dirichlet Allocation (LDA)**[10] (from SKLearn), with the document-term matrices created using a CountVectorizer. The dataset consists of just under **250** songs, about half (123) of which are by **Eminem**, and the rest are by **Drake** and **The Script**.

n_topics = 3

3 is agreeably not an ideal value even for such a small dataset. However, I intend to make a point through it. Let's look at some of the top words for two of the three topics; two of the words that were a bit too ungracious have been omitted from the image below:



Words for Topic 2 (with n_topics = 3)



Words for Topic 3 (with n_topics = 3)

Here's the Wheel of Emotions (WoE) for reference: (image at wikipedia) (png)

According to the WoE, there are 8 primary bipolar emotions: joy versus sadness; anger versus fear; trust versus disgust; and surprise versus anticipation. The primary emotions can be expressed at different intensities (moving to & away from the center) and can mix with one another to form different emotions.

An interesting thing to note is while emotions such as joy and sadness are bipolar in the WoE, a song may express both.

Both the images above suggest, *and this is a bit of an oversimplification (but then again, so was choosing 3 topics & a such small dataset in the first place)*, that '**love**' is one of **the generalized emotions** the two topics represent, and that's not just because the top word is love, the other words matter as well. Also, it seems that words for Topic 3 have comparably **sadder undertones**.

Speaking of undertones, while not a fact, it could be said that Eminem's songs usually have a *baseline* of **anger** present in them. The dataset majorly consisted of songs by Eminem, and lyrics to those are usually laced with profanity. That is why Topic 1's words are not pictured here, as they were heavily influenced by such lyrics from Eminem (and the others as well, to some extent). What I am trying to imply is that the major emotional connotation that the words of that topic carried was anger (and negative emotions in general).

I am not implying that the Topic represents Eminem's songs by itself, though I would say it does a considerable portion. One such Eminem song that is not necessarily represented by that topic is Mockingbird. The lyrics to this song express his **love** for his family, **sadness** over how some things have happened and are going on, **regret** over how some things have turned out to be and his inability to change that, with possible undertones of **anger**. The topic association of this song according to the model is:

Topic 1	Topic 2	Topic 3
0.45362375	0.00209083	0.54428542

This *seems* to (somewhat?) line up. However, it wouldn't necessarily for all songs. While that could be difficult to achieve, the trick would be to find the number of topics that enables the model to do something like this for a majority of the dataset.

I would now like to hark back to an earlier point in the document where I said each of the songs would be assigned emotions that are associated with the ***topics*** that are most prevalent in the song. The plurality here is important. While it may be a good idea to associate the songs with emotions from just the most prevalent topic, a better approach would be to assign a union of the more prevalent topics instead. This specially becomes a more suitable approach as the number of topics increase, as that would cause an increase in the specificity of emotion in each topic. In case of the above example, I would go with the union of the emotions from topics 1 & 3.

There is another approach that I considered, but decided not to go through with: instead of assigning a simple '**yes**' or '**no**' ('1' or '0') for an emotion being expressed by a song, it is possible to **take the (sum of the) prevalence values of the topics in the song as the**

values of the emotions the topics express. However, this ends up making the second problem a regression problem, and I currently wish to stay with classification.

This section has been longer than I expected. I would like to end it by homing in on my mention of increase in specificity with increase in number of topics.

n_topics = 4

While the other two topics (from the n_topics = 3 model) remain more or less the same, the one topic that did not have its words pictured in this document is divided into two subtler

topics when the number of topics is increased to 4. In the picture above, topic 1 resides in the realm of anger and aggression while topic 2 hinges on rage. The topics get more subtler on increasing n_topics, but I would like not to extend this section any longer.

Solution Statement

The solution for the first part would be to create a Topic Model, preferably using **LDA**. I would experiment with models & tools available in **SKLearn, TensorFlow, Gensim, and nltk** (and others) and the parameters of those models until I find an appropriate number of topics and get a satisfying clustering result. Another small example of this (using **MALLET**[11]) is included in Project Design. After clustering, each topic will then be associated with about 1-5 emotions, which may be mutual or exclusive, which in turn will allow the songs to be assigned emotions as well.

The second part of the problem is a multilabel classification problem for which I would create a deep neural network and use TensorFlow to help me out where necessary. I have been studying ways to use CNNs and RNNs for text classification and may (or may not) go down that road.

Benchmark Model

I have not considered using an evaluation metric for the first problem

The Benchmark Model for the second problem would be the **Multi-layer Perceptron classifier (MLP)**[12][13]. The final model being designed by this project is essentially going to be an extension of MLP in some way, specially if the final model does not turn out to be Convolutional or Recurrent (though in a way those could also be considered so), and happens to be a simple, though still deep, neural network.

Evaluation Metrics

For this section, **n_topics** = number of topics, the parameter for the Topic Model.

While I will consider values of the **Silhouette Score** and perhaps the **Dunn Index** for different n_topics, the final decision would be based on **how diverse the topics end up being**. I would look at a fixed number of top words for each topic for different values of n_topics, and base my decision on that when choosing n_topics. It is possible that the metrics line up with the choice, but because of the subjectivity and the soft nature of the clustering, that may or may not happen.

For the more objective problem, I would go with the **F-Score**[14][15] (with **β**=0.5) as the metric over other **multilabel classification metrics**[16] such as the **Hamming Loss**[17][18] or the **Jaccard Index**[19][20]. There are two reasons for choosing $\mathbf{F_{0.5}}$:

- I would like to favor precision over recall
- Label imbalance is likely*, and can be accounted for using the F-Score.

It is also possible that I may perform classification on a somewhat specifically (algorithmically) selected subset of the Topic Modeled dataset to attempt to manually account for some of the label imbalance (if it is there), but I may not be able to do so efficiently, hence F-score.

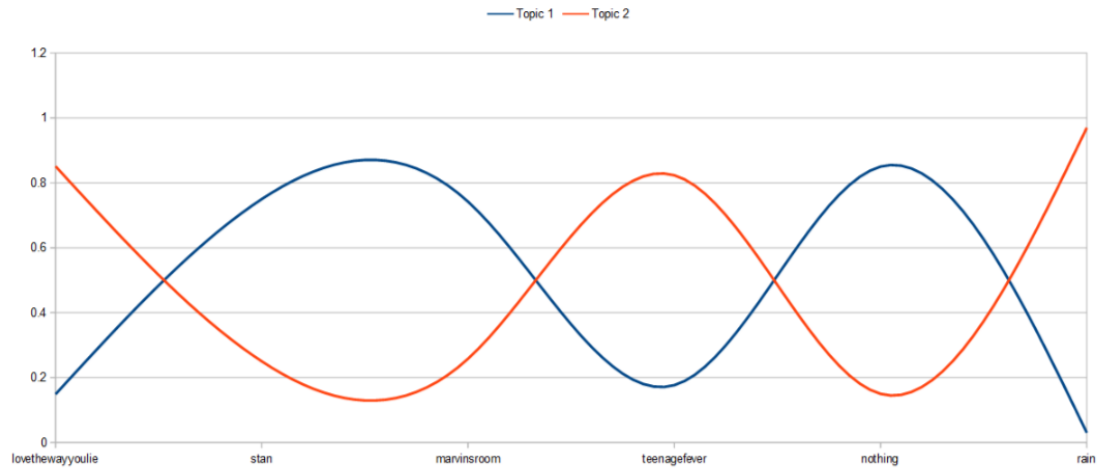
** data dependent, as seen in the dataset section, when n_topics = 3, Topic 1 had the majority of the terms*

Project Design

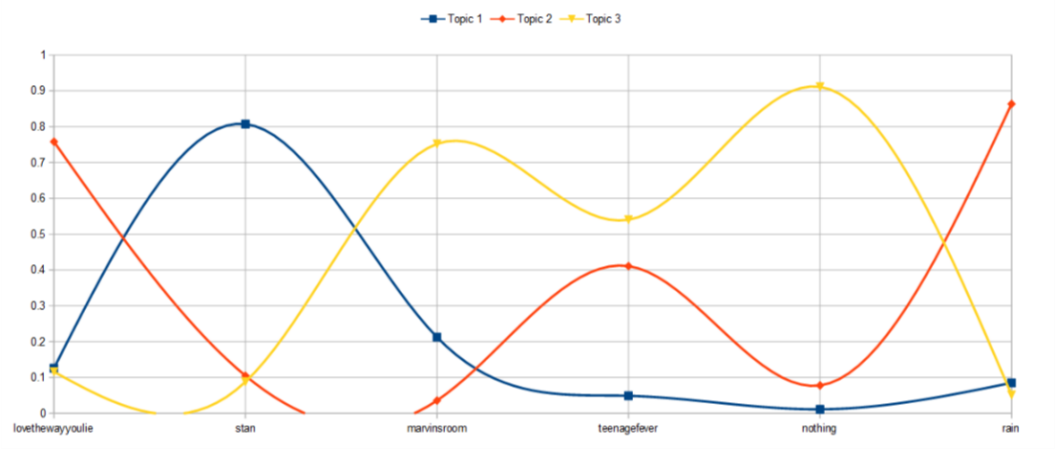
The first step would be **cleaning** - this would involve making the text lowercase and then using **regular expressions** to remove unnecessary symbols, then using **nltk** to remove **stop words**, and perhaps **stemming**.

The second step would be to extract the **integer word counts** if using **LDA** or converting the words into **tf-idf**[21] format if using **Latent semantic analysis (LSA)**[22], whichever is the correct format for the model selected from one of the available libraries.

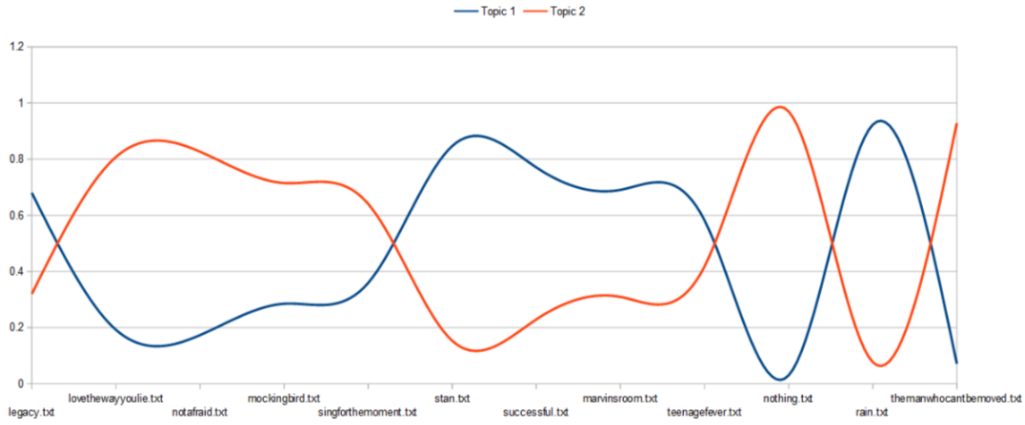
One such library, mentioned earlier, is MALLET. Below are some results of topic modeling using MALLET on lyrics. I wish the images were clearer here. I will try to describe them the best I can. Zooming may make the text readable if it isn't.



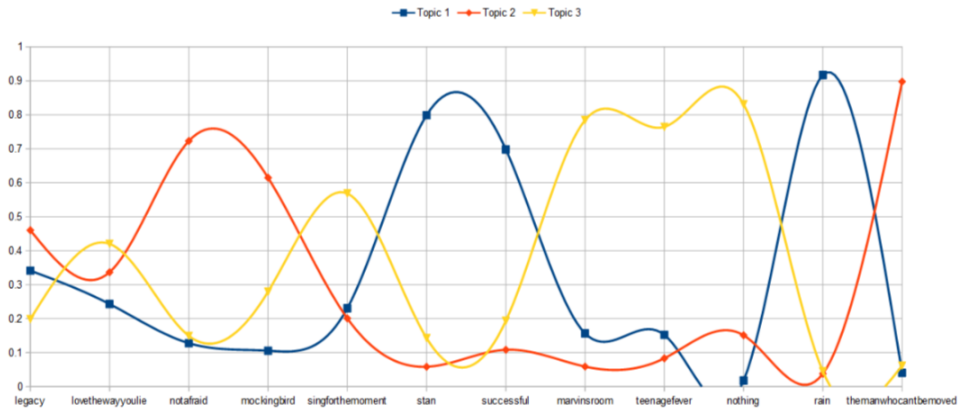
6 Songs, 2 Topics



6 Songs, 3 Topics



12 Songs, 2 Topics



12 Songs, 3 Topics

The first two images model 6 songs into 2 & 3 topics respectively. The following two images do the same for 12 songs. The key takeaways that I intend to portray from these results are:

- How a change in the dataset or the number of topics affects the model's output:
    - When there are 6 songs, both cases (n_topics = 2, 3) put **Marvin's Room** and **Nothing** into the same cluster. Either case has one topic that relates to words such as "drunk, call, phone, etc.". In case of n_topics = 2, this ends up putting **Stan by Eminem** into the cluster as well, while increasing n_topics causes other subtleties of Stan to be recognized and its cluster to change.
    - When the number of songs is changed to 12, and more lyrics come in, the clustering is visibly affected. For example: Marvin's Room and Nothing are separated when n_topics = 2, and in the same cluster when n_topics = 3.
- How topic modeling can help with cleaning of the dataset:
    - In the 6 songs, 2 topics case, **Rain by The Script** acts as somewhat of an anomaly as one of the top keys of the topic that is majorly portrayed by that song (according to the model) happens to be a repeated 'oh'. Without elaborating too much, what I intend to say here is that looking at the top words within a certain topic can be helpful in identifying whether it is being defined by words that do not necessarily carry meanings for the song or are repeated one too many times.
    - Some artists (especially in Hip-Hop music) tend to use their own name, in some cases multiple times in the same song. The first visualization in this document (using pyLDAvis) removed words like 'Eminem', 'Marshall', 'Mathers', 'Shady', 'Drake', etc. before vectorizing the text.
    - Such anomalies may carry less weight as the dataset increases and more meaningful words take part in the creation of the topics. Their removal will thus be decided contingent upon how the model turns out to be with and without them.
    - Identification of contractions that are slang and not necessarily taken care of by stopword removal and stemming, such as 'tryna' = 'trying to'.
    - Lastly, I would also like to see how much change results in the model after removal of (some or all of the) profanity. I know in the dataset section I oversimplified things by pairing it with anger, but as things get more nuanced, we start seeing other behaviours / patterns. The final topic model would most likely not have them removed, but I would like to see the results before I make that decision.

The next step is assigning emotions to the topics (and transitively, to the songs). By the time I finish implementing the topic model, I would also have finalized the set of emotions to use.

With the output of the topic model ready and the emotions finalized and assigned to the dataset, the words from the lyrics would then be, if they haven't already been, **stemmed,** and then converted into the tf-idf format, **split into train, validation and test sets**, and fed first into the benchmark model (MLP), and then into the final model for training.

The design of this final model has not been finalized yet, as I am still studying the various text processing and classification techniques, and the math behind them. Whichever method I end up using, the final model will be a deep neural net that would perform multi-label classification with (vectorized) lyrics as the input and a set of emotions as the output for a given song.

I really do wish that I could have been more specific here, but I do not currently know how things would turn out to be (would I use a CNN or an RNN?, for example). I would be delving deeper into them as soon as the Topic Modeling part of the project is over. Whatever the case may be, I hope it turns out to be awesome!