Analyzing Rap Lyrics – Part 2: Preparing, Cleaning, and Exploring Rap Lyrics

Posted on **March 9, 2016**

To briefly recap, the end result of Part 1 was the creation of a corpus of lyrics based on the weekly Billboard (BB) 15 Hot Rap Songs and extracted from the ChartLyrics.com archive using their API. On closer review of the lyrics, it turns out that a number of the songs were either missing, contained major errors or omissions, or were some language other than English. As a consequence, where possible I replaced the missing or erroneous songs with corrected versions from Genius.com, resulting in a final tally of 1206 tracks (which are available in the Dataset section of this blog).

In their current state, this corpus of lyrics is basically a collection unstructured text that requires a series of steps to convert into a structured format that is amenable to data analysis and mining. It's these steps that are the focus of this part of the analysis.

From Unstructured to Structured Data

Most data analysis and data mining techniques rest on the assumption that the data are structured. Structured data "refers to any data that resides in a fixed field within a record or file." For example, most of the data in relational databases and spreadsheets qualify as structured. The fact that these data rest on an explicit data model and are well-formed and organized makes them amenable to numerical and analytical manipulation. In contrast, unstructured data "refers to information that either does not have a pre-defined data model or is not organized in a pre-defined manner." It's typically, although not exclusively, "text-heavy" which in its native form requires transformation before numerical techniques can be applied.

Simply stated, before text can be manipulated or mined it needs to be "turned into numbers" which can then be analyzed with various computational, statistical and data mining methods. As Jennifer Thompson (2012) suggests,

The purpose of Text Mining is to process unstructured (textual) information, extract meaningful numeric indices from the text, and, thus, make the information contained in the text accessible to the various ... statistical and machine learning algorithms. Information can be extracted to derive summaries for the words contained in the documents or to compute summaries for the documents based on the words contained in them. Hence, you can analyze words, clusters of words used in documents, etc., or you could analyze documents and determine similarities between them or how they are related to other variables of interest in the data mining project.

Here, the word document is a catchall term that represents the individual entities making up a corpus. For a corpus of tweets it's the single tweet, for blogs it's the individual blog, and for our rap lyrics it's the individual song track.

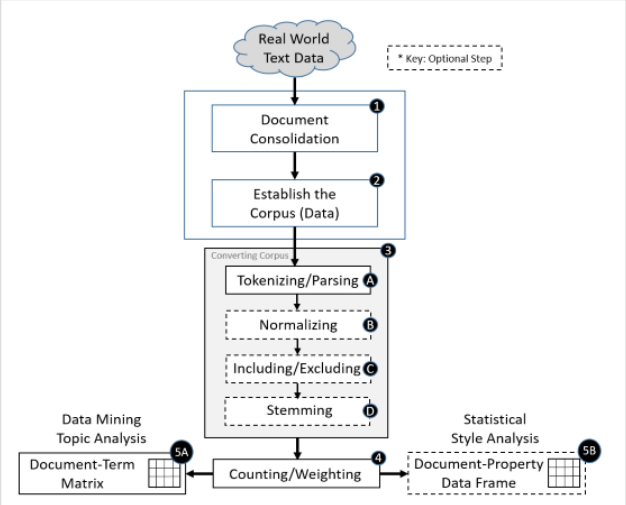

Figure 1. Steps for Transforming Text to Structured Data

In the case of text (like our corpus of rap lyrics), the transformation is accomplished with the aid of various computational linguistics and statistical natural language processing (NLP) techniques. As Figure 1 shows (at the bottom), the specific steps that are used in transforming text into numbers depend in part on the type of analysis or mining to be performed.

In prior research about the classification of song lyrics (see Appendix 1), two types of analysis have generally been performed:

1. *Content* – This type of analysis focuses on the content in the lyrics, determining the similarities or differences in the specific words or sequences of words among the individual songs or groups of songs in the corpus. As indicated in Figure 1, this is done by using natural language processes to convert the text in the corpus of lyrics into individual terms or sequences of terms (S-3) to determine both the unique terms found in the corpus (i.e. the *vocabulary*), as well as how frequently each of the terms or sequences occurs in the individual songs or groups of songs (step 4).

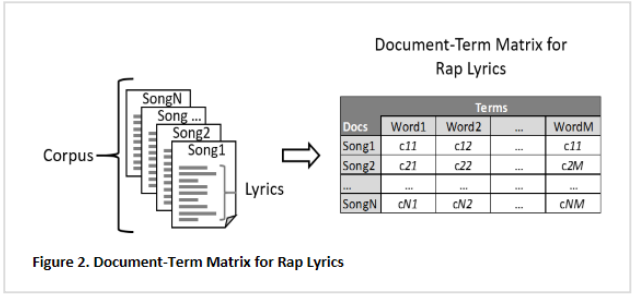

Figure 2. Document-Term Matrix for Rap Lyrics

Usually these steps produce a *document-term matrix* (S-5A) where the rows represent the individual documents or a group of documents, each column represents one of the unique terms or sequences of terms in the corpus, and each cell of the matrix — the intersection of

row-column — indicates the number of times (weighted or unweighted) that each term occurs in the document (see Figure 2). In mining this corpus of lyrics, we end up mathematically comparing the row vectors of numbers in the matrix with one another – often basing the comparison on the computed distances among the vectors — in order to discover the underlying topics found within the various songs or groups of songs.This sort of matrix is also called a "Bag of Words (BOW)." It's the type of structure that MXM used to represent their million song dataset. This type of structure is not without its critics. As Yang and Chen (2011) point out, lyrics differ from regular documents such as news articles in a variety of ways:

- Lyrics are composed in a poem-like fashion with rich metaphors that make word disambiguation difficult and degrade the performance of unigram models (like BOWs).
- Negation terms such as *no* or *not* can play important roles in lyric analysis, especially those focused on emotion and sentiment. Simply treating them as single words or as stop words can alter the semantic meaning completely.
- Lyrics are recurrent because of stanzas. This recurrent nature is not modeled by BOWs since word order is disregarded.

For this reason, many of these critics have focused on the *structure* and *style* embodied in the lyrics rather concentrating on their *content*.

2. *Style* – While the content of a lyric refers to "what was written or said," the *style* refers to the "way it was said." As Fell and Sporleder (2014) note, songwriters employ unique stylistic devices to build their lyrics, some of which can be measured automatically, and some of which are unique enough to distinguish one song or group of songs from another. Examples of these devices include the use of non-standard words and slang, rhythmic structures, references to the past and present, and self- or other-referencing (e.g. "I" versus "you). Essentially, this type of analysis begins with a set of predefined features and associated categorical or ordinal values along with a dictionary (listing) of specific terms, phrases or (grammatical) structures that represent those values (i.e. include lists for S-3C). For example, words or phrases that are slang (versus non-slang) or specific grammatical structures that represent various types of rhyming. With these features and values in hand, the task is to convert the lyrics in a song or group of songs into individual words, phrases or structures that enable automatic matching and counting of the (absolute or relative) number of times particular values within the dictionary occur (S-4). In this type of analysis, standard statistical tests (e.g. Chi-Square, F-tests, and ANOVA) are used to compare the songs or groups of songs.Although there is no explicit requirement in this type of analysis to generate the equivalent of a document-term matrix (S-5B), sometimes it makes programmatic sense to do so.

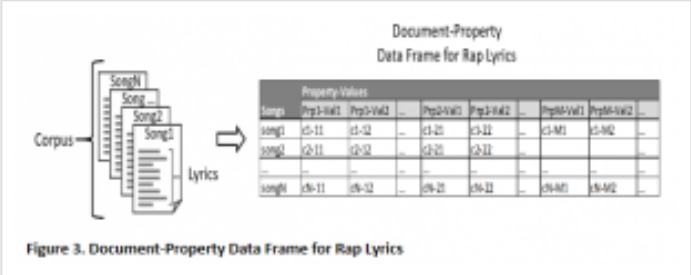

Figure 3. Document-Property Data Frame for Rap Lyrics

One option is to use a structure like the one created by Pennebaker et al. and used in an application known as the Linguistic Inquiry and Word Count (LIWC). Here, the documents are still the rows, but the columns are the values of one or more properties, and the row-column intersection represents the number of times (absolute or relative) that a particular property value occurs within that document (see Figure 3). So, in our case, the rows are the songs or groups of songs, the columns might include slang words, swear words, past-tense verbs, etc., and the intersections would be the (absolute or relative) number of slang words, followed by the number of swear words, followed by the number of past-tense verbs, etc. This type of table is called a *data frame* in the programming worlds of R, Python, and Spark.

In practice, it's not really a binary choice between the doc-term matrix or the doc-property data frame. Instead, those studies that focus on *style* rather than *content* usually perform a BOW analysis in order to have a baseline against which to compare the classification capabilities of the various stylistic properties.

Converting the Corpus

For our purposes Steps 1 and 2 in Figure 1 have are already been completed since we have an "established corpus" of rap lyrics already in hand. The next Step (3) involves a series of mini-steps aimed at "converting" the corpus. These mini-stems are based on a standard natural language processing (NPL) and include:

- *Step 3A – Tokenizing*: Parsing the text to generate terms or tokens. Sophisticated analyzers can also extract various phrases and structures from the text. This is typically a first step in the conversion process regardless of the type of analysis being performed.
- *Step 3B – Normalizing*: Using some combination of filtering and conversion capabilities to discard punctuation, eliminate numbers, and to convert uppercase letters to lowercase. With the exception of eliminating numbers, analysis aimed at understanding the *content* usually embraces all of these steps. In contract, analysis focused on the *style* typically ignores this step with the exception of converting all the terms to lowercase. When you making everything lowercase, the capitalized and lowercase versions (e.g. "Shorty" and "shorty") will be treated and counted as the same word rather than as two.
- *Step 3C – Including or Excluding Particular Classes of Words*: The heart of *style* analysis revolves around a series of *include lists* (dictionaries) against which the tokens produced in Step 3A are compared. In contrast, the analysis of *content* revolves around a series of *exclude lists* or cutoff points. For example, take a look at Figure 5. It shows the counts for the 50 most frequent lower case tokens (words) in our sample of 1206 rap lyrics. Notice anything about the tokens or words on the horizontal axis? These are not the words that come immediately to mind when you think of rap lyrics. These are all words we use constantly in everyday discourse. The overwhelming majority of them can be classified as *function* words, as opposed to *content* words. In NLP vernacular they are also called *stopwords*. These are the words that "connect, shape, and organize content words." There is no way to avoid them in written or spoken communication. In English there are a little over 300 of these words, yet this small number usually accounts for 50% or more of all the words in any kind of corpus. Since every song has a plethora of these kinds of words, they are of little value in distinguishing the content from one song to the next. So, they are eliminated.

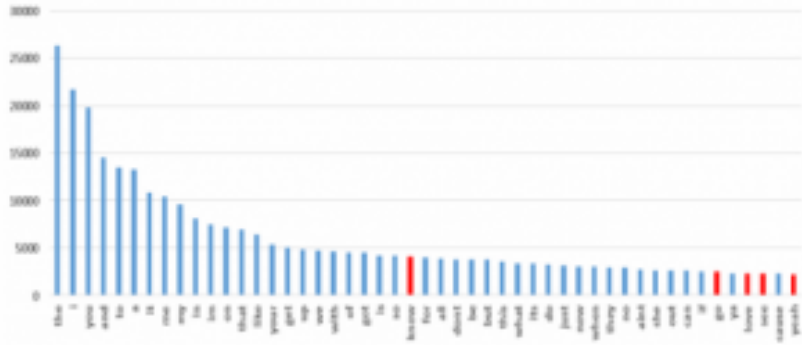

Figure 4. Top 50 Tokens or Words in Rap Lyrics by Frequency of Occurrence (1211 Lyrics) - red denotes content words

In the analysis of *content*, the same exclusion tactic is often used at the other end of the spectrum to eliminate words that occur very infrequently. Those words that occur only once in an entire corpus are called *hapax legomenon* (*hapax* for short). What happens if you exclude one of these tokens — say "357" (that's a gun), or "skitzomanie", or "blooooood", or even "blueprint" all of which are hapax in our sample of rap songs? That answer is not much. However, what happens if we exclude not one but all the hapax? Figure 7 tells the story. In our sample of lyrics, it turns out that there are around 660K tokens in the corpus (excluding punctuation), while there are about 23K tokens in the vocabulary underlying the corpus. If you eliminate all the hapax (the tall blue bar on the left), you knock out 55% of the vocabulary (~13K/23K) but only 2% (23K/660K) of the corpus. If you eliminate those tokens that appear 2-4 times in the corpus, you knock out another 28% of the vocabulary and only 3% more of the corpus. When comparing one song or group of songs with another, the elimination of these infrequent words are not likely to have much of an impact on any topic modeling process. This is one of the reasons that this strategy was used to build the BOWs for the MXM million song dataset. Recall that the MXM data set started with a corpus of 55M words with an underlying vocabulary of ~500K unique words. Of the 500K words, 238K were hapax and another 112K occurred between 2-4 times. Eliminating these still left a handy 111K words, way too many for analysis. In the end, the vocabulary was whittled down to the 5000 most frequently occurring stems (not words) including the stems for the function. This left 92% of the original corpus (51M out 55M tokens).Again, this sort of exclusion is antithetical to the analysis of *style*. Eliminating all these infrequent words can substantially impact, for example, the counts for specialized groups of words like *echoisms* — songs with repeating characters or substrings — or rhyming sequences.

- *Step D – Stemming:* The final step in the conversion processing is stemming which involves converting a token or word into its root form so that noun forms like 'doggs' and 'dogg' are considered the same rather than unique, or verb forms like "want', 'wants', or 'wanted' are treated as the same rather than as different words. The result of the stemming process is often an shortened word form (e.g. so that 'achieve', 'achieves', 'achieved' and 'achieving' all become 'achiev'). Converting words or tokens to their stemmed form is an art rather than a science – a crude one at best. For English language text, there are a variety of stemming algorithms that can be used to carry out the task. By far the most popular is Porter's, which is what I used with my sample of rap lyrics.Stemming is basically a chopping algorithm, so it doesn't handle word forms that mean the same thing but are spelled differently or word forms that are spelled the same but mean different things. This is where *lemmatization* comes into play. This process removes inflectional forms (e.g. 'wants' to 'want') and returns the base

form (i.e. lemma) of a word so, for example, that the verb forms 'am', 'are', and 'is' would yield 'be'. In the same vein it can distinguish words that are spelled the same but play different roles in a particular context (e.g. the verb 'saw' which is converted to 'see' versus the noun 'saw' which is converted to 'saw'). Unfortunately, in the world of rap lyrics both standard lemmatization and stemming fall far short because of the large number of slang terms, abbreviations and intentional misspellings (e.g. most verbs ending in 'ing' are shortened to 'in'). In essence they're fine when faced with standard English. They don't work well when the text strays from the standard path.Because the lemmatization and stemming algorithms produce very similar results (especially after the stopwords are removed), I haven't applied formal lemmatization to the lyrics. I did, however, create special functions to handle swear words and some of the slang terms. In Rap, swear words and their close relatives have an amazing variety of spellings and presentations. For example, in my sample of Rap lyrics the word m***f***r occurred a little over 700. There were over 60 different renditions of the word with the standard dictionary spelling accounting for about 200 occurrences. To handle these specialized terms and versions, I created conversion routine to transform them into the same dictionary spelling or a substitute term (for presentation purposes).As you might imagine, stemming is a no-no in the analysis of style. It would make it impossible, for example, to determine the tense of any verb.

Converting Rap Lyrics

Simple Example

With these general steps in mind, let's look at how the conversion steps work for the lines from the end segment of a rap song in the corpus called "Ease My Mind" by Arrested Development (1994). The choice of this particular song and segment is basically random, although the wording is more straightforward than most of the other 1205 songs in the collection and misogyny and vulgarity are virtually non-existent (so on the surface the words don't automatically offend anyone).

Lyric Segment
... Answer my prayer to give my Earthly Body inner inner peace. Until that day upon which my souls released! need some time need some time when the sista's say it then you know it's gotta be real! Thrusted in a world that I don't know from my mommas lips between my mamas hips I'm cuddled by her hands because she understands It's that bond that will keep the movement movin' on It's that bond that will keep the movement movin' on. And so I GOTTA BE READY TO RELINQUISH gotta be ready to relinquish, my breath of life for struggle. That is why I tell you... I need some time to ease my mind.
3A - Tokenize: ['Answer', 'my', 'prayer', 'to', 'give', 'my', 'Earthly', 'body', 'inner', 'inner', 'peace', '.', 'Until', 'that', 'day', 'upon', 'which', 'my', 'souls', 'released', '!', 'need', 'some', 'time', 'need', 'some', 'time', 'when', 'the', "'sista's", 'say', 'it', 'then', 'you', 'know', "'it's", 'gotta', 'be', 'real', '!', 'Thrusted', 'in', 'a', 'world', 'that', 'I', "don't", 'know', 'from', 'my', 'mommas', 'lips', 'between', 'my', 'mamas', 'hips', "I'm", 'cuddled', 'by', 'her', 'hands', 'because', 'she', 'understands', "It's", 'that', 'bond', 'that', 'will', 'keep', 'the', 'movement', 'movin', "'", 'on', "It's", 'that', 'bond', 'that', 'will', 'keep', 'the', 'movement', 'movin', "'", 'on', '.', 'And', 'so', 'I', 'GOTTA', 'BE', 'READY', 'TO', 'RELINQUISH', 'gotta', 'be', 'ready', 'to', 'relinquish', ',', 'my', 'breath', 'of', 'life', 'for', 'struggle', '.', 'That', 'is', 'why', 'I', 'tell', 'you', '...', 'I', 'need', 'some', 'time', 'to', 'ease', 'my', 'mind', '.']
3B - Normalize - Alphanumerics & Lowercase: 'answer', 'my', 'prayer', 'to', 'give', 'my', 'earthly', 'body', 'inner', 'inner', 'peace', 'until', 'that', 'day', 'upon', 'which', 'my', 'souls', 'released', 'need', 'some', 'time', 'need', 'some', 'time', 'when', 'the', 'sistas', 'say', 'it', 'then', 'you', 'know', 'its', 'gotta', 'be', 'real', 'thrusted', 'in', 'a', 'world', 'that', 'i', 'dont', 'know', 'from', 'my', 'mommas', 'lips', 'between', 'my', 'mamas', 'hips', 'im', 'cuddled', 'by', 'her', 'hands', 'because', 'she', 'understands', 'its', 'that', 'bond', 'that', 'will', 'keep', 'the', 'movement', 'movin', 'on', 'its', 'that', 'bond', 'that', 'will', 'keep', 'the', 'movement', 'movin', 'on', 'and', 'so', 'i', 'gotta', 'be', 'ready', 'to', 'relinquish', 'gotta', 'be', 'ready', 'to', 'relinquish', 'my', 'breath', 'of', 'life', 'for', 'struggle', 'that', 'is', 'why', 'i', 'tell', 'you', 'i', 'need', 'some', 'time', 'to', 'ease', 'my', 'mind']
3C - Exclude Stop and Function Words: ['answer', 'prayer', 'earthly', 'body', 'inner', 'inner', 'peace', 'day', 'souls', 'released', 'time', 'time', 'sistas', 'gotta', 'real', 'thrusted', 'world', 'mommas', 'lips', 'mamas', 'hips', 'cuddled', 'hands', 'understands', 'bond', 'movement', 'move', 'bond', 'movement', 'move', 'gotta', 'ready', 'relinquish', 'gotta', 'ready', 'relinquish', 'breath', 'life', 'struggle', 'time', 'ease', 'mind']
3D - Stemming (Porter): [u'answer', u'prayer', u'earthli', u'bodi', u'inner', u'inner', u'peac', u'day', u'soul', u'releas', u'time', u'time', u'sista', u'gotta', u'real', u'thrust', u'world', u'momma', u'lip', u'mama', u'hip', cuddli', u'hand', u'understand', u'bond', u'movement', u'move', u'bond', u'movement', u'move', u'gotta', u'readi', u'relinquish', u'gotta', u'readi', u'relinquish', u'breath', u'life', u'struggl', u'time', u'eas', u'mind']

Figure 6. Converting the Lyrics for "Ease My Mind
[note: the u'word notation stands for the unicode output from the stemming algorithm]

The lines from the segment are shown in Figure 6 along with the results of the various conversion steps. The results were produced by a Python program I created using the *natural language toolkit (nltk)* in combination with a collection of regular expressions and some input/output code modeled after earlier work done by Toby Segaran (2007). In other text analysis, I've also relied sklearn feature extraction module in Python and on a text mining package (TM) in "R" to accomplish similar sorts of tasks. They all can do the job.

Impact on the Rap Corpus

Term	Corpus	Vocabulary
3A - Tokenize	760883	33354
3B - Normalize	671814	24395
3C - Exclude	256977	23238
3D - Stemming	242169	17824

Table 1. Total Number of Terms in the Rap Corpus and Vocabulary for each Conversion Step

With each step, the number of terms in the corpus and the associated vocabulary diminishes. The reduction is apparent in Figure 6, but it's readily seen in the aggregate counts that result when the entire corpus of rap lyrics is converted (see Table 1). Here, it's obvious that the biggest impact occurs when the stop and function words are excluded.

As noted, the need for specific steps varies depending on the type of analysis – *style* versus *content*. Those studies focused on *style* usually perform some form of tokenization and normalization and end up working with the lowercase versions of the tokens. In contrast, most studies dealing with *content* perform steps 3A-3C, while step 3D is optional. For instance, in his study of the evolution of the content in pop lyrics, James Thompson (2014) chose to bypass stemming. In his words,

After scraping the lyrics, I then built a Python script to process them into a useful format… This involved clearing out punctuation, rogue characters etc. then splitting the lyrics into words and removing the stop words…The Python stop words package provides a good list. I also experimented with stemming; which takes the word back to its root stemming becomes stem *etc. In the end I preferred the results without this…*

which means that his LDA analysis employed the *terms* that resulted from excluding various stop and function words (the *non-stops*).

I'm actually performing both types of analysis. For *style*, I'll be working primarily with the lowercase terms. For *content*, I'll be using the stems which is in keeping with the MXM BOWs

Creating a Song-Stem Matrix: Counting and Weighting

It's a bit more straightforward to understand the process of creating a *document-term matrix* (a BOWs) to be used in topic modeling, as opposed to creating *a document-property data* frame for analyzing style. So, let's start with the BOWs or bag-of-stems (BOS) in this case.

Selecting the Stems (Columns)

Recall, that the end goal for this type of lyric analysis is a matrix whose rows are songs (i.e. documents), whose columns are the unique stems (i.e. terms) across all the songs, and the cells represent the number of times that a particular song contains a particular stem. If it doesn't contain the stem, the number in the cell is zero (0). There are a variety of ways to come up with the list of unique stems for a corpus, but basically we gather all the stems from all the songs in the corpus and find the "set" for this list. Mathematically, a set is a collection of distinct or unique elements. In our vernacular it's the vocabulary.

Sorted Set of Stems: [u'answer', u'bodi', u'bond', u'breath', u'cuddli', u'day', u'earthli', u'eas', u'gotta', u'hand', u'hip', u'inner', u'life', u'lip', u'mama', u'mind', u'momma', u'move', u'movement', u'peac', u'prayer', u'readi', u'real', u'releas', u'relinquish', u'sista', u'soul', u'struggl', u'thrust', u'time', u'understand', u'world']

Figure7. Assorted Set for Stems in "Ease My Mind"

For example, the (sorted) set associated with the list of stems at the bottom of Figure 6 is shown in Figure 7. Here, the list of stems has gone from 42 to 32. When you gather the stems for the entire corpus into a single list and then create a set from the list, the aggregate change corresponds to the numbers at the bottom of Figure 8. The list of stems has ~242K terms, while the set of unique stems is ~18K terms. Practically speaking, 18K elements is too large for analytical purposes, after all the MXM BOW is only 5K and it represents the corpus for 210K songs versus the 1206 rap songs in my corpus.

As the earlier discussion (S-3C) suggested, one way to reduce this number is to eliminate terms that appear very infrequently in the corpus. While you can do this earlier in the process, I decided to postpone this step until after I had empirically examined the results from stemming. It's akin to what James Thompson did during his topic analysis of pop lyrics:

Now I needed to build the LDA (topic) model… The biggest decisions I needed to make were around how many iterations to run, and how many topics I wanted to find. I ran the process with a variety of iterations… It was during this process I also discovered the real art to LDA is in the stop words. I found myself adding many more to the list, particularly more lyric based 'noises' rather than words. Things like oohhh, ahhh, doo deee, dah *etc. These weren't adding any value to my topics.*

Of course, it's probably easier to look at a frequency distribution of all the terms or stems to determine what constitutes "noise" rather than excluding them in a piecemeal fashion. As

discussed in S-3C, one way to find the "noise" is by determining those terms or stems that only appear a few times in the corpus, say 4 times or less. Figure 5 only hints at the impact of this reduction because it's based on the tokens not the non-stops or stems. Using the stems, if you eliminate those that occur 4 times or less, the number of unique stems drops about 70% and the corpus of stems by about 40%. This still leaves a list of close to 5000 unique stems, which may be too large a number for the sample size of songs (i.e. 1206 rows versus 5000 columns).

A second way to eliminate the "noise" is by looking at those terms or stems that only appear in a small percentage of the songs, say 10% or less. This is the sort of tactic suggested by Segran (2007). In fact he suggested excluding those terms that appeared in less than 10% of the documents (songs) on the low end and more than 50% on the high end. For this analysis, the high end has already been covered pretty much simply by eliminating the function or stop words. So, the stem that appears in the most songs is "time" which shows up in 54% of the songs. If we jettison those that appear in 10% or less of the 1206 songs, we'd end up with ~220 unique stems and reduce the corpus of stems by ~40%. Here, 220 stems may be too small.

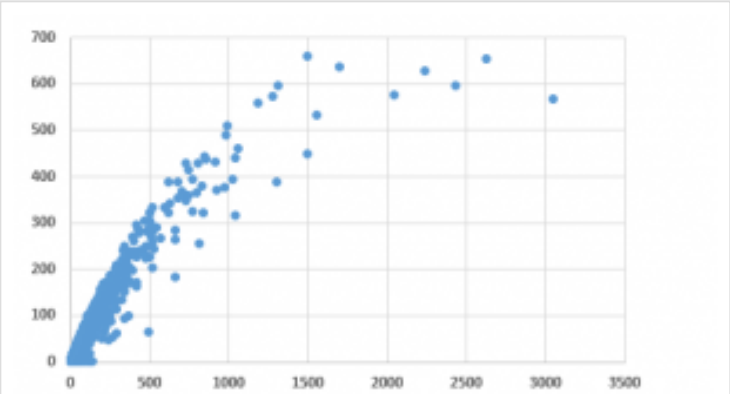

Figure 8. Number of Songs that a Stem Appears in Versus Frequency in the Corpus

As Figure 8 shows, the number of songs in which a stem appears is strongly related to the number of times it appears in the corpus (obvious). If they we're linearly related, the correlation "r" would be .92. The choice is kind of a tossup although they're easily combined.

Somewhat arbitrarily, I decided for starters to work with those stems that occur a minimum of 50 times in the corpus and appear in at least 50 songs. This combination yields about 500 unique stems (note: Figure 9 provides a look at the Top 50 of the remaining stems – those that appear most frequently in the corpus). So, the starting song-stem matrix has 1206 rows and 500 columns. I say starting because it will take a few iterations through the topic modeling process to determine whether 500 is sufficient to distinguish the songs and to determine whether the topics have evolved over time.

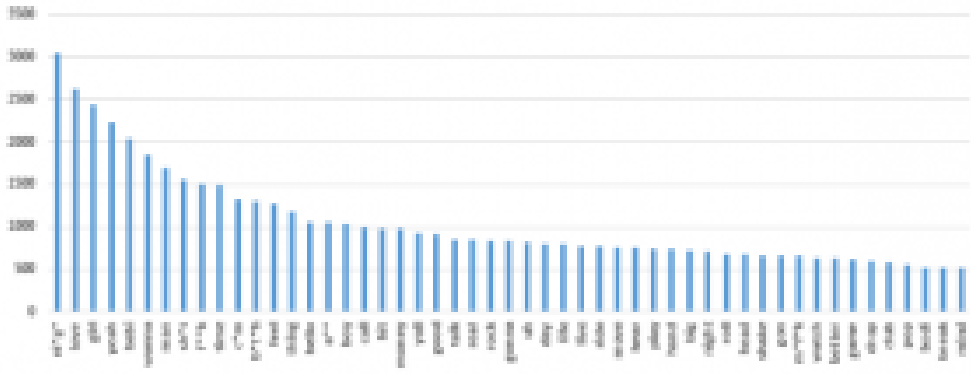

Figure 9. Top 50 Stems that Appear Most Frequently in the Corpus

Weighting the Counts

As it now stands, each of the cells in the song-stem matrix indicates the actual number of times that a particular stem occurs in a particular song. When determining the importance of a word within a song or comparing the differences in word counts from one song to another, the raw number can be inflated because some songs have more words than others and some words occur more frequently in the corpus than others. For these reasons, It's standard practice in text analysis to substitute a weighted value for the raw count. A common weighting scheme is *term frequency-inverse document frequency* (tf-idf). There are various ways to compute tf-idf, but the simplest version is:

tf-idf = tf(t,d) * log(D/Di)

where *tf* is the frequency of term t in document *d*, *D* is the number of documents, and *Di* is the number of documents containing the term.

As Manning et al. (2008) note, td-idf is:

1. highest when the term or stem occurs many times within a small number of documents (thus lending high discriminating power to those documents);
2. lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
3. lowest when the term occurs in virtually all documents

For example, the stem "time" appears in more songs than any other stem — 661 out of the 1206 songs. In the song "Party to Damascus"(Wyclef Jean 2003) it appears 6 times. So, instead of using a 6 for this cell in the matrix, we'd use the weighted value 3.6 (i.e. tf-idf = 6 * ln(1206/661)), reducing its prominence in the song because the stem occurs so frequently in the corpus.

Song-Stem Matrix for Topic Modeling

	Stems			
Songs	Stem1	Stem2	...	Stem500
song1	tdidf(1,1)	tdidf(1,2)	...	tdidf(1,500)
song2	tdidf(2,1)	tdidf(2,2)	...	tdidf(2,500)
...	...	...	...	...
song1206	tdidf(1206,1)	tdidf(1206,2)	...	tdidf(1206,500)

Figure 10. Song-Stem Matrix with Weighted *td-idf* Values

The end result of the various conversion, selection and weighting steps outlined in Figure 1 is a song-stem matrix like the one shown in Figure 10.This is the form that will be used for topic modeling in Part 3 of the Analysis.

Creating a Document-Property Data Frame for Analysis of *Style*

Lyrical style covers a wide range of properties – everything from emotions and moods, to the intelligence factor, to the rhyming structure. Among the various studies concerned with lyrical *style*, the analysis by Fell and Sporleder (2014) deals with the widest range of properties. They group the various properties into five dimensions and 13 features including:

- Vocabulary — TopK n-grams (n<=3), lexical diversity (type-token ratio) and non-standard words
- Style – POS/chunk tags, length, echoisms, and rhyme features
- Semantics – Conceptual imagery
- Orientation – Pronouns, past tense
- Song Structure – Chorus, title and repetitive structures.

In this analysis I'll be covering a subset of these dimensions and features including:

Dimension	Feature	Measures for Each Song in Corpus
Vocabulary	Lexical Diversity	No. of Lowercase Voc./No. of Lowercase Terms
	Uncommon Words	No. of Lowercase Terms not appearing in Wiktionary
	Slang Words	No. of Lowercase Terms appearing in Urban Dictionary
Style	Length	Lines per song, Lowercase Terms per song, Lowercase Tokens per line
	Rhyming	Rhyming Factor computed from Rapalyzer Algorithm (Malmi 2015)
	Echoisms	No. of Lowercase words with repeating letters (3 or more) and/or repeating substrings
Semantics	Conceptual Imagery	Inventory of Psychological and Social Processes and Personal Concerns ascertained by the Linguistic Inquiry and Word Count (LIWC)
Orientation	Temporal	Ratio of past tense verb forms to present and future tense (LIWC)
	Egocentricity	Ratio of 1st person pronouns to 2nd and 3rd person(LIWC)

Table 2 . Dimensions, Features and Measures Incorporated in Statistical Style Analysis

The measures for the first two dimensions in the above subset — vocabulary and style — are very similar to those used by Fell and Sporleder. The only exception is that I'm using a newly constructed algorithm recently posted by Eric Malmi (2015). For the other two dimensions — semantics and orientation — the features are generally the same but the measures are more extensive. The reason why it's more extensive is not because I've personally come up with a

better set of measures. It's because I'm relying on a computer application called the Linguistic Inquiry and Word Count (LIWC). LIWC is the result of years of research by James Pennebaker et al. that began back in the early 1990s. The basic premise of all this work is that "the words we use in daily life reflect who we are and the social relations we are in," as well as "what we are paying attention to what we are thinking about, what we are trying to avoid, how we are feeling, and how we are organizing and analyzing our worlds." Towards this end, they have developed LIWC, an application that analyzes written (and spoken) text with respect to 90 psychometric properties. For the most part Pennebaker et al. have used LIWC for a "higher purpose" — determining and improving an individual's psychological well-being — as opposed to using the application to analyze rap lyrics. Fortunately, the LIWC is very simple to use, yields a much broader coverage of the semantic and orientation dimensions than previous studies, and can be applied to virtually any corpus.

Song-Property Matrix for Statistical Style Analysis

	Features			
Songs	Voc/Terms	Uncommon Wrds	...	Past/Pres-Future
song1	v/t(1)	uwcount(1)	...	p/p-f(1)
song2	v/t(2)	uwcount(2)	...	p/p-f(2)
...	...	...	...	...
song1206	v/t(1206)	uwcount(1206)	...	p/p-f(1206)

Figure 11. Song-Feature Data Frame of Style Features

Although it is certainly not mandatory, once the data corresponding to the features and measures in Table 2 have been completed, it's useful to put them in a data frame which can then be used for further analysis. For example, this is what the LIWC does. It's also what I've done for the next stages of the statistical analysis of *style*.

Prelude to Part 3

As many researchers have lamented, in most data analysis projects the processes of gathering, cleaning and preparing the data occupy about 80% of the time. The other 80% (that's not a typo) is devoted to the actual modeling, communicating, visualizing, reporting and packaging. It's these later steps that are discussed in Part 3 of this series.

Resources

References

Bertin-Mahieux, T. et al. "The Million Song Dataset." *International Society for Music Information Retrieval* (2011).

Fell, M. and Sporleder, C. "Lyrics-Based Analysis and Classification of Music." 25th *International Conference on Computational Linguistics* (2014).

Manning, C. and Schutze, H. Foundations of Statistical Natural Language Processing. MIT Press (1999).

Manning, C., Prabhaker, R. and Schutze, H, and . Information Retrieval. Cambridge University Press (2008).

Tausczik, Y. and Pennebaker, J. The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods. Journal of Language and Social Psychology (2010). [Note: there are a number of articles by Pennebaker et al. This has one of the more complete descriptions of LIWC].

Segran, T. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reily (2007).

Thompson, James. "The Evolution of Popular Lyrics and a Tale of Two LDA's." (2015).

Thompson, Jennifer. "Text Mining (Big Data, Unstructured Data." (2012).

Yang, Y. and Chen. H. *Music Emotion Recognition (Multimedia Computing, Communication and Intelligence) 1st Edition*. CRC Press (2011).

People

Christopher Manning

Toby Segran

James Pennebaker

Tools and Modules

Linguistic Inquiry and Word Count (LIWC)

Python Natural Language Toolkit (NLTK)

Python Sklearn Feature Extraction Module

R Text Mining (TM Package)

This entry was posted in **Uncategorized** by **daveking63@gmail.com**. Bookmark the **permalink [http://dataffiti.com/2016/03/09/analyzing-rap-lyrics-part-2-preparing-cleaning-and-exploring-rap-lyrics/]** .

ONE THOUGHT ON "ANALYZING RAP LYRICS – PART 2: PREPARING, CLEANING, AND EXPLORING RAP LYRICS"

youtube
on **October 15, 2016 at 5:24 am** said:

It's really a cool and helpful piece of info. I am glad that you just shared this useful information with us.
Please stay us informed like this. Thanks for sharing.

Comments are closed.