Creating a Hip Hop Graph

Rafal Nowak
therafalnowak@gmail.com

2018-05-30

Outline

01 Introduction/the TL;DR
02 Motivations
03 Hip Hop NLP Problem
04 Making the Graph
05 Analysis, Extensions, Discussion
06 Conclusion

01 What are we looking to create here?

Now here's a funky introduction of how nice I am
Tell your mother, tell your father, send a telegram

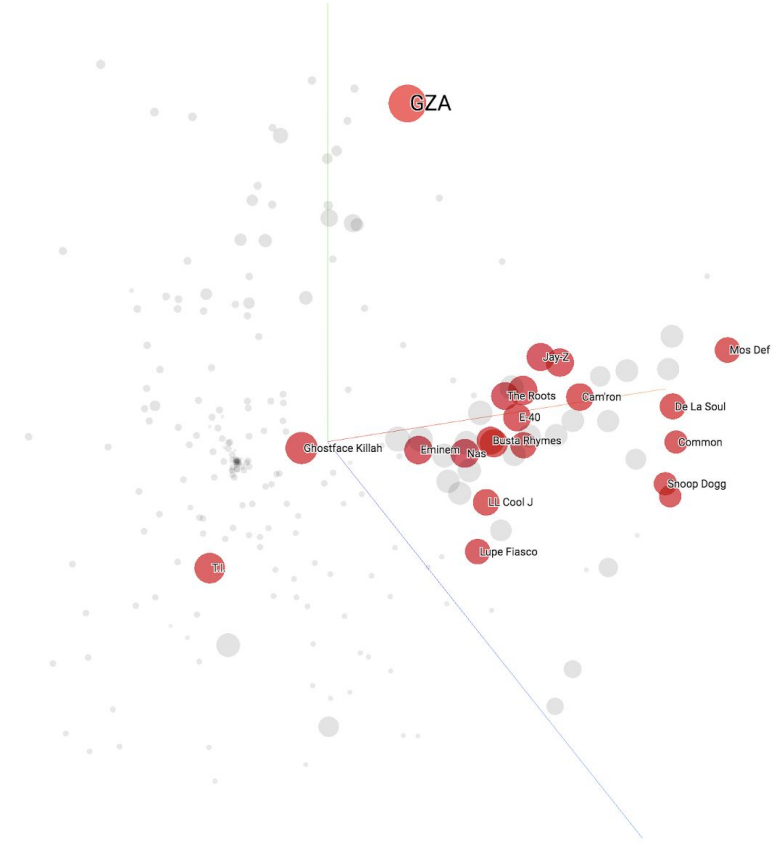
Check the Rhime
A Tribe Called Quest - Low End Theory 1991
Universal Music Publishing Group

Introduction

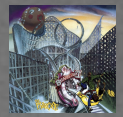
TL;DR

- Utilizing ideas from web structure, search algorithms, and graph theory, we analyze lyric data for a set of hip hop artists to find the language most endemic to the genre and most endemic to each artist in the genre through a technique called TF-IDF (term frequency - inverse document frequency)
- We find artists that are most alike by computing the cosine similarity between artists' TF-IDF vectors
- We run into a dimensionality problem when computing similarity and creating a visualization, so we employ the Sparse Singular Value Decomposition algorithm (SVD-S) to reduce the language vector dimensionality
- To create our final graph, we employ a machine learning visualization algorithm - t-distributed stochastic neighbor embedding (t-SNE) - within a Tensorflow projector
- We then analyze the graph for meaning and discuss its possible use cases through the lens of *the Spotify algorithm*
- Lastly, we restate what we've already stated - that makes a good conclusion!

Final Product (2- and 3-D)



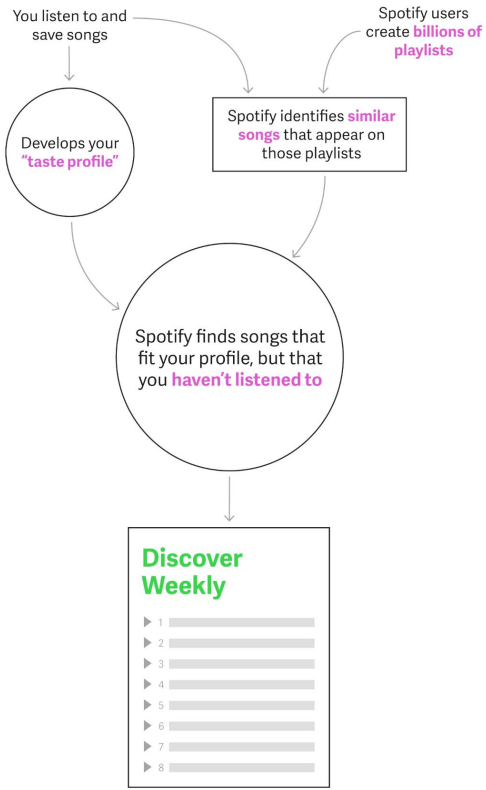
It took a second to register up in my branium
My dome, my head, my skull, my cranium
My eyes have had enough, it was time to do some talking
I had to creep through the hound-dogs that were stalking



Otha Fish
The Pharcyde - Bizarre Ride II The Pharcyde 1992
Sony/ATV Music Publishing LLC

Motivations

Spotify (or like) Music Recommendation Algorithms



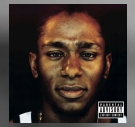
- Discover weekly based on user-generated playlists
- How do you learn from user feedback?
- What features do you include?
  - Acousticness, Danceability, Energy, Instrumentalness, Liveness, Speechiness, Tempo, Valence
- Products that require learning (user-level) and feedback:
  - Discover Page
  - Discover weekly playlists
  - Radio (which is *the infinite* playlist)
  - Related Artists

The Language of Hip Hop

- an article from *The Pudding*

- Main motivations, methodology, and data came from an article from *The Pudding*, a data visualization publication
- The main data and algorithm steps in this presentation are an effort to replicate the work done in The Pudding article
- The focus of this presentation is ***how*** to create a graph in the first place
- Graph analysis through theory and extensions of this model will be discussed at a high level at the end

The Mighty Mos Def..
It's simple mathematics
Check it out!
I revolve around science..
What are we talking about here?



Mathematics
Mos Def - Black on Both Sides
Sony/ATV Music Publishing LLC

Hip Hop NLP

Data Sourcing and Aggregation

- Artist - song - lyric data came from LyricFind in “bag of words” format
- Dataset has the lyrics for 276k songs along with some metadata (artist, song title, song id, word id, lexical mapping, etc.)
- Data is extracted from text files, uploaded to a Hadoop environment and aggregated to create a master dataset for analysis
    - Data steps include aggregating multiple metadata tables and unnesting the bag of words. Song-word level data consisted of 67m rows, after counts - 27m.
- 238 Hip hop artists within dataset are labeled with a boolean variable, rest are assumed non-hip hop

Computing the Odds - the “most hip hop” words

- Output: list of the 30 most and least hip hop words
- Odds are: likelihood the word is in the catalog of a hip hop artist vs. the rest of the corpus
- Computation:

$$\frac{\text{Number of times word appears in the hip hop corpus}}{\text{Total number of words in the hip hop corpus (~36k)}} = \text{hip hop word proportion}$$

$$\frac{\text{Number of times word appears in the non-hip-op corpus}}{\text{Total number of words in the non-hip-hop corpus (~65k)}} = \text{corpus word proportion}$$

- Divide the two proportions and we have the odds a word appears in the hip hop corpus.
- Sort odds descending and we have *most* hip hop words
- Sort ascending gives *least* hip hop

The “most hip hop” and “least hip hop” words

snoop	pimping	thug
balling	bitches	hustling
hoe	jay	hood
nigger	motherfucking	motherfuckers
hoes	slang	weed
niggers	rims	stacks
hustle	projects	beef
homey	rep	gat
pimp	pussy	bitch
rappers	gangsta	bricks

someone	alone	happy
falling	beautiful	sweet
heart	feels	may
were	na	inside
sun	along	sing
tears	loving	found
away	such	above
has	moment	seems
lies	dreams	cry
wind	dream	singing

Computing the most endemic words by hip hop artist

- Each hip hop artist gets assigned a Term Frequency Inverse Document Frequency (TF-IDF) score for each word in the hip hop corpus of words
- TF-IDF computation: count a word comes up in the catalog of an artist (TF) is divided by the number of artists that have used that word at least once (IDF) to make TF-IDF
- Method required sub-linear scaling: essentially take (1+log(TF)) instead of TF, making this in essence a "weighted" TF, or WF
- Computation yields: (1+log(TF)) / IDF, and the words with the highest TF-IDF scores are deemed the *most endemic* to that particular artist

The most endemic words per artist (subset of artists)

2pac	A Tribe Called Quest	Eminem	GZA	RZA	Wu-Tang Clan	Xzhibit
mourn	muhammad	aah	members	swords	pretending	chucks
outlaw	motivate	guzzle	clan	stallion	tang	gangsters
screamed	tribe	stan	strictly	digital	meth	sliding
ers	quest	aftermath	crash	tang	failed	gators
insanity	award	bizarre	metal	bobby	sword	golden
cemetery	dang	creek	rhyme	sword	method	pimps
untouchable	queens	penis	with	jag	bees	approach
currency	vibe	darling	mic	lab	clan	bail
released	stressed	dave	from	bee	allah	steady
hopeless	rhythm	royce	not	bob	ending	arm

Computing the similarity between artists

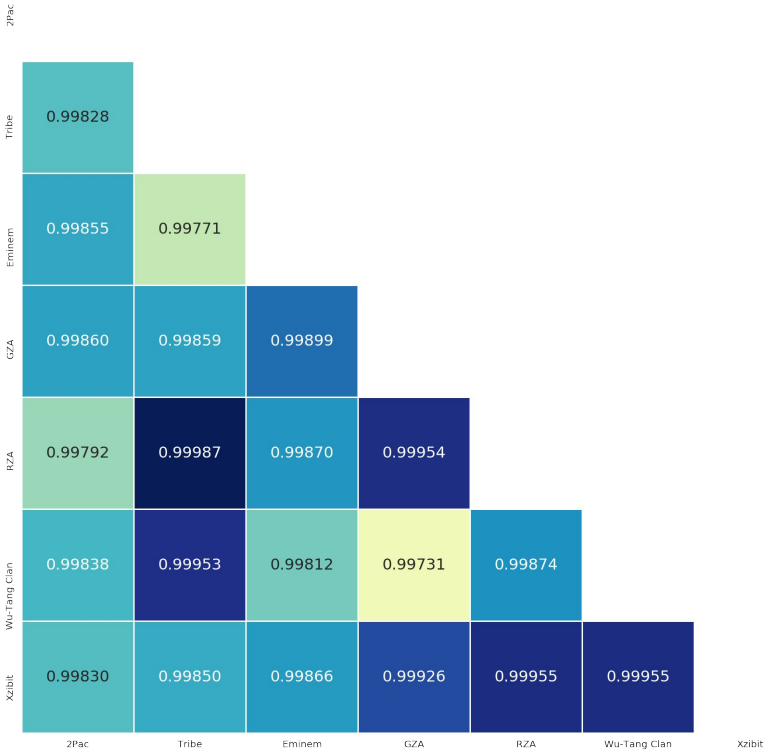
- Cosine similarity is often used in information retrieval and comparing scores between documents
- It is a measure of similarity between two non-zero vectors of an inner product space that measures cosine of the angle between them
- Cosine of 0 is 1, and then anywhere between 0 and pi/2 is between 0 and 1
- It is derived from the Euclidean dot product formula: $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$
- Cosine similarity formula, where A and B are two separate artist vectors

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Other methods of computing similarity: Jaccard index (similarity or dissimilarity between sample sets, think Venn Diagram), RBF Kernel (common in machine learning), and Euclidean distance (norms, straight line)

Cosine similarity for a set of artists

- The cosine similarities between a set of artists creates a symmetric matrix
- The heat map indicates the cosine similarity for a subset of artists
- RZA and A Tribe called Quest are most similar (highest cosine similarity), while GZA and Wu-Tang are least similar
    - But GZA is a former member of Wu-Tang and we would assume they would be similar
- The cosine similarities are close to 1, meaning nearly identical, reason being the dimensionality is super high: 36,218 to be exact (one for each distinct word in the hip hop language)
- We have a dimensionality problem



Another dimension, another dimension
Another dimension, another dimension
Another dimension, another dimension
Another dimension, another dimension
Another dimension, another dimension
Another dimension

Intergalactic
Beastie Boys - Hello Nasty 1998
Capitol Records

Dimensionality Problem

Sparse Singular Value Decomposition (SVD-S)

- To tackle the dimensionality problem, we take the full WF-IDF matrix and perform SVD-S
- SVD is a dimensionality reduction algorithm useful in many applications
- Since our matrix is largely sparse (lots of zero-valued WF-IDFs for many dimensions), we employ a variation of SVD-S (for Sparse matrices). Despite its name, it works fine for dense matrices as well
  - Uses the ARPACK Python iterative procedure to find the SVD decomposition by reducing the problem to an eigendecomposition on the array dot(X.T, X)
  - Regular SVD is much faster, but since it uses LAPACK, it requires a lot more memory (scaled with the number of dimensions)
  - Final result: computes the 50 largest k singular values/vectors in our sparse matrix

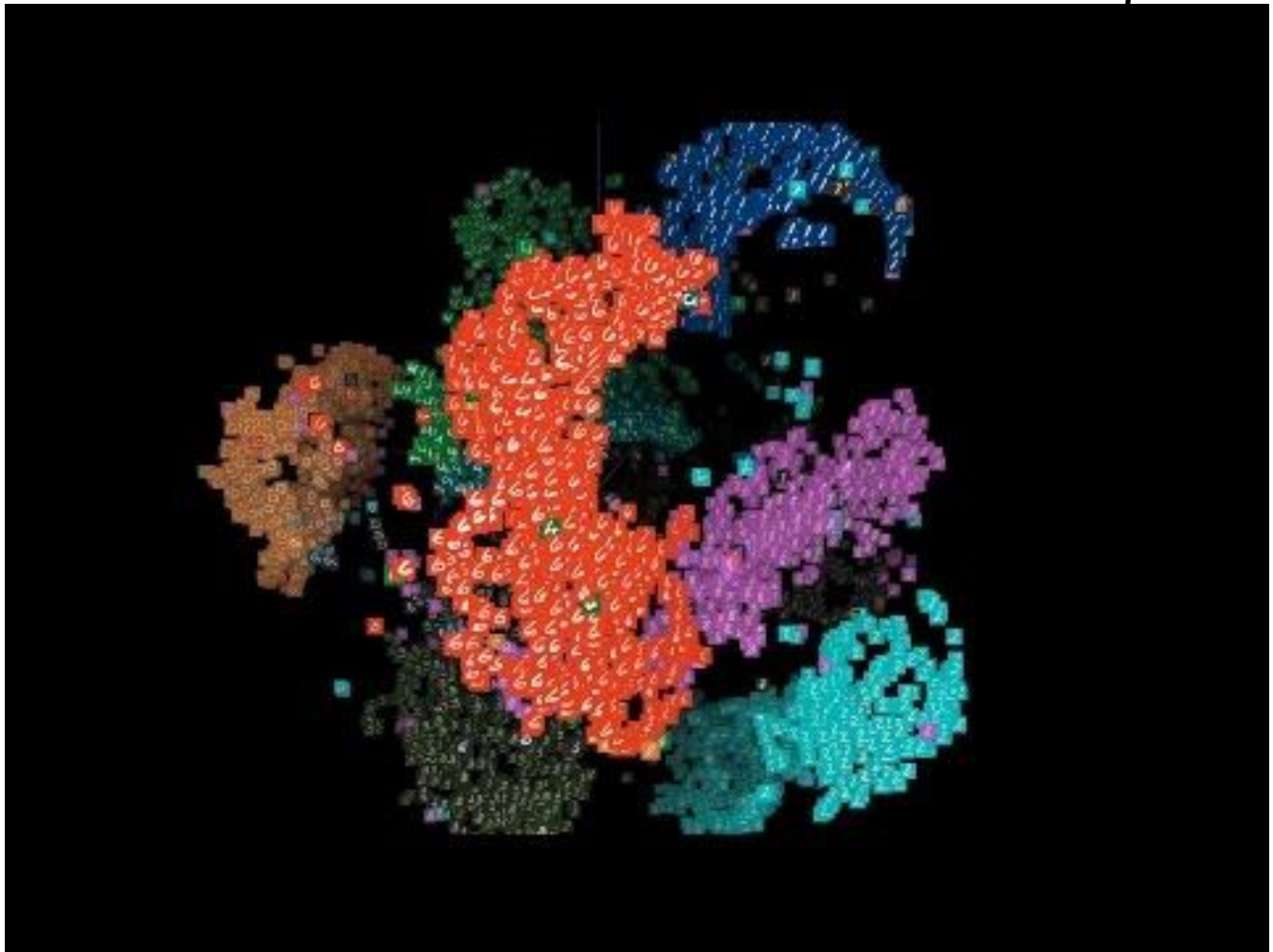
t-Distributed Stochastic Neighbor Embedding (t-SNE)

- It is a variation of Stochastic Neighbor Embedding, reducing the tendency to crowd points in the center of the graph and good for reducing to 2- or 3-dimensions

- Two steps to the algorithm (technical):
  - 1 - algorithm creates probability distributions between high dimensional objects in such a way that similar objects have a high probability of being picked, while dissimilar points have an extremely small probability of being picked
  - 2- algorithm defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map
    - Kullback–Leibler divergence is a measure of how one probability distribution diverges from a second, expected probability distribution

- The algorithm is non-linear and adapts to the underlying data, performing different transformations on different regions
  - uses random walks on neighborhood graphs to allow the implicit structure of all of the data to influence the way in which a subset of the data is displayed
- Generally unreliable for clustering since distances are not preserved

t-SNE visualization in Tensorflow Projector

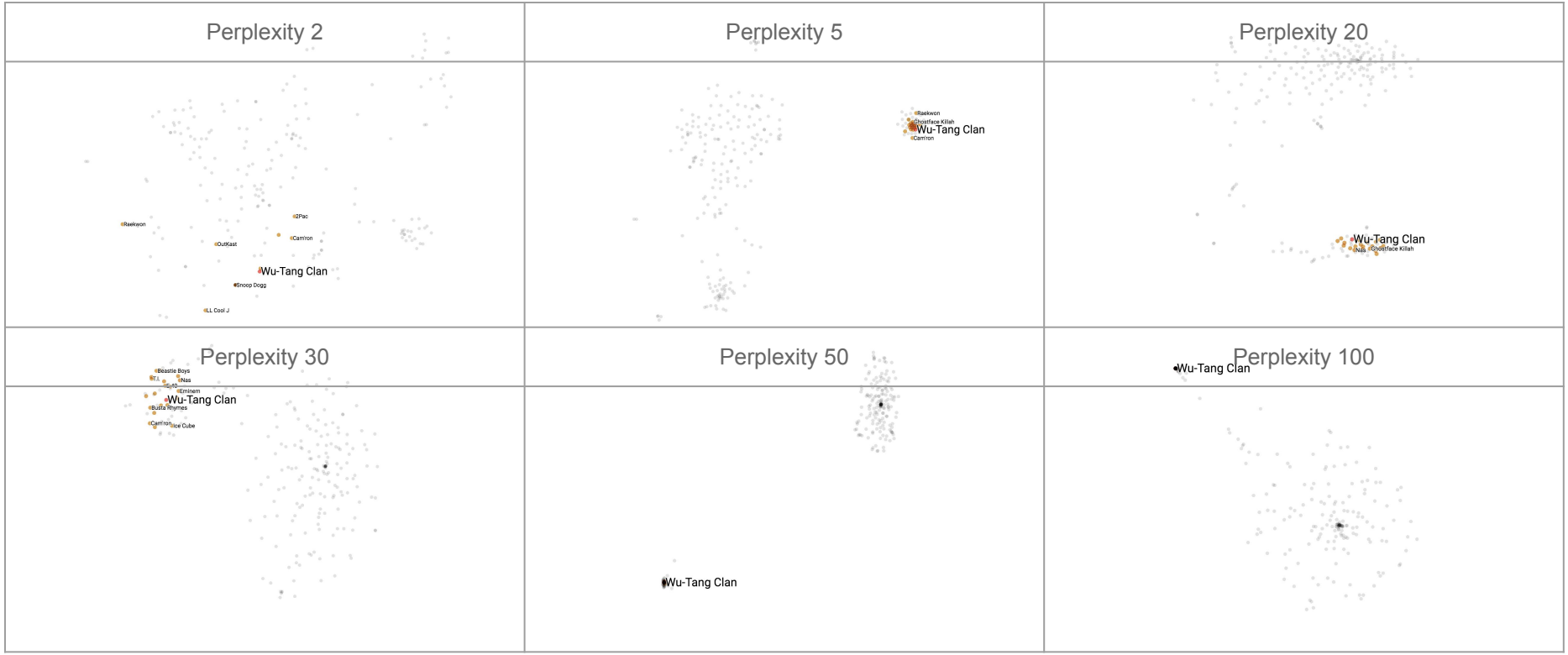


t-SNE Tensorflow Projector tune-able parameters

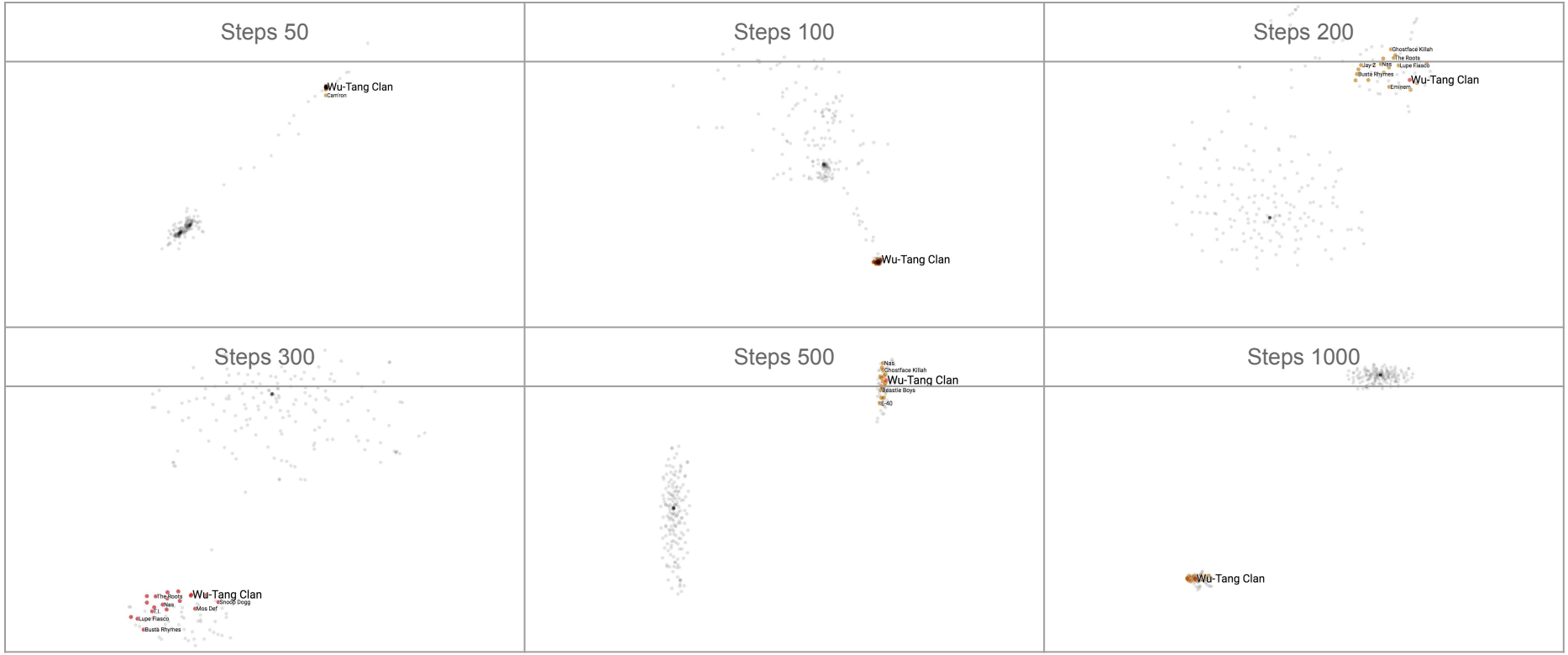
- Data: SVD-S Vectors and Metadata
- Other projector inputs:

	Dimension	Perplexity	Learning Rate (Epsilon)	Iterations
What is it? (loosely)	Select 2- or 3-d visualization	Balances attention between local and global aspects of your data. It is a guess about the number of close neighbors each point has.	The extent newly acquired information overrides old information	Number of times the two steps of the algorithm are computed
What are the inputs?		{2,3,...,100}	{0.001, 0.01, 0.1, 1, 10, 100}	{1,2,...,inf.}
Best Practices		Depends on the density of the data - dense data requires a larger perplexity	Depends on the size of the data, smaller datasets having a slower learning rate	Depending on the data, as needed

Constant Epsilon (10) and Steps (500) Variable Perplexity



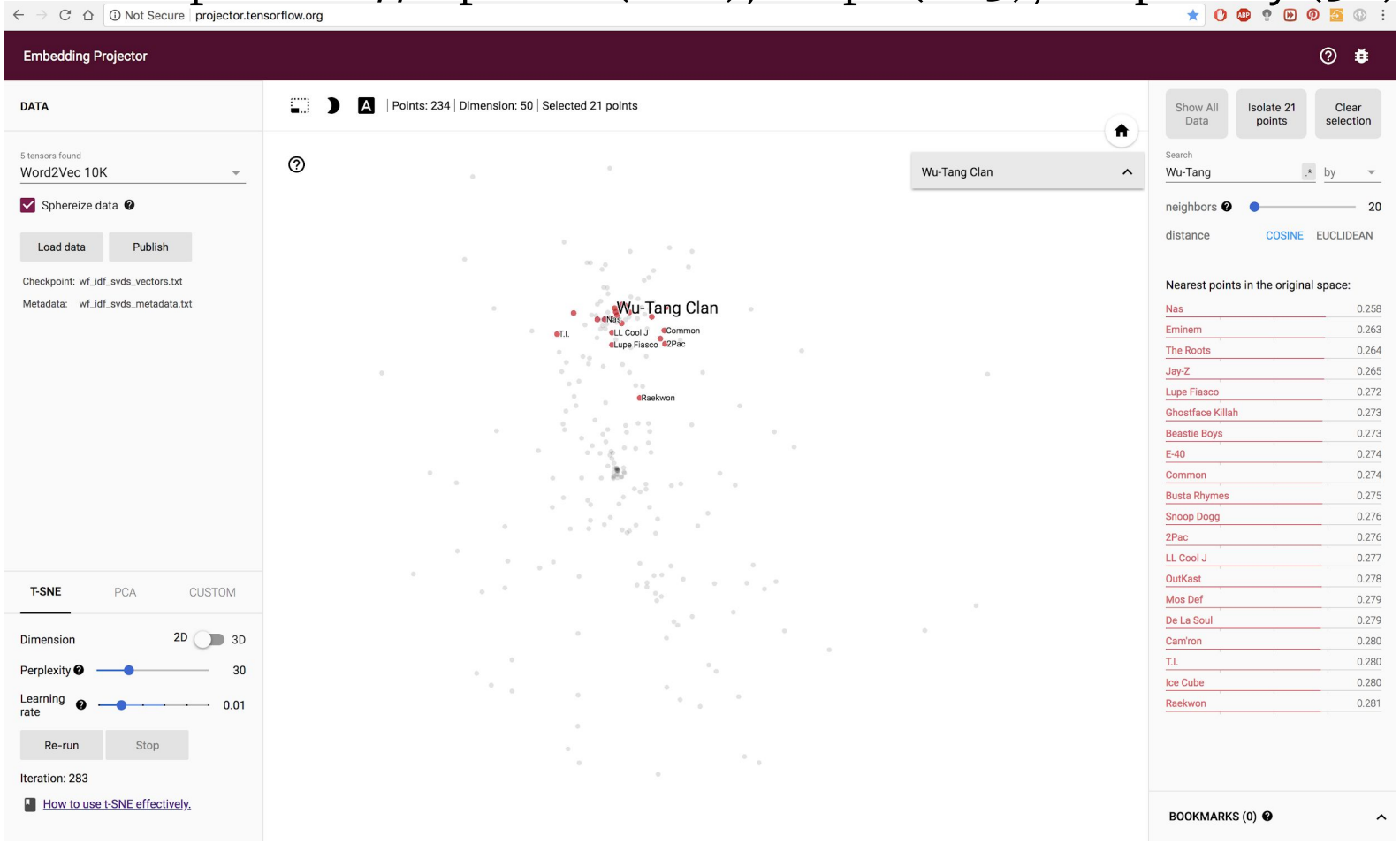
Constant Epsilon (10) and Perplexity (30) Variable Steps



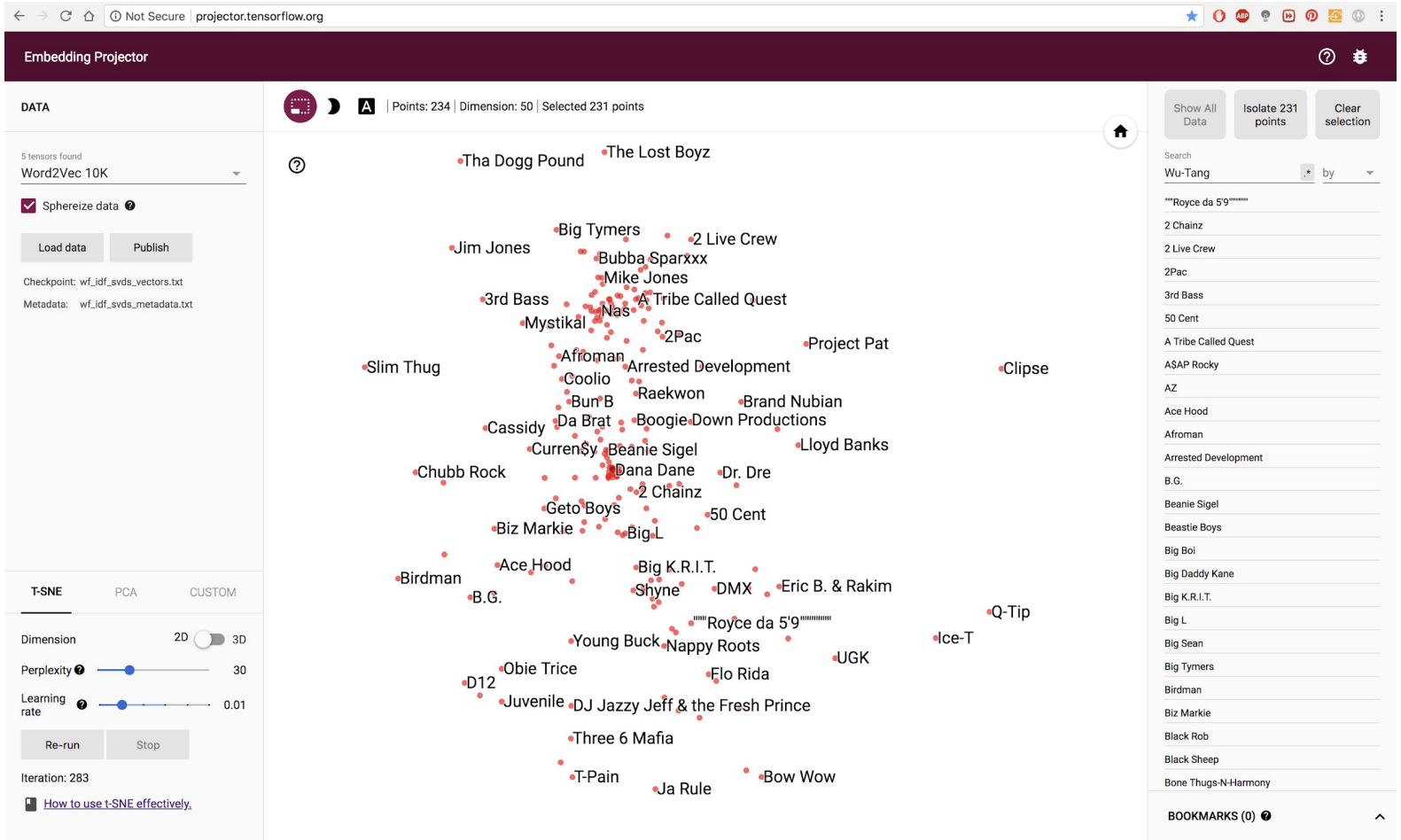
Things to keep in mind with t-SNE

- Hyperparameters matter
  - There are other hyperparameters related to the optimization process, so successive runs in the projector may not yield similar results
- Cluster sizes in a t-SNE plot mean nothing
- Distances between clusters might not mean anything
- Random noise doesn’t always look random
- You can see some shapes, sometimes
- For topology, you might need more than one plot

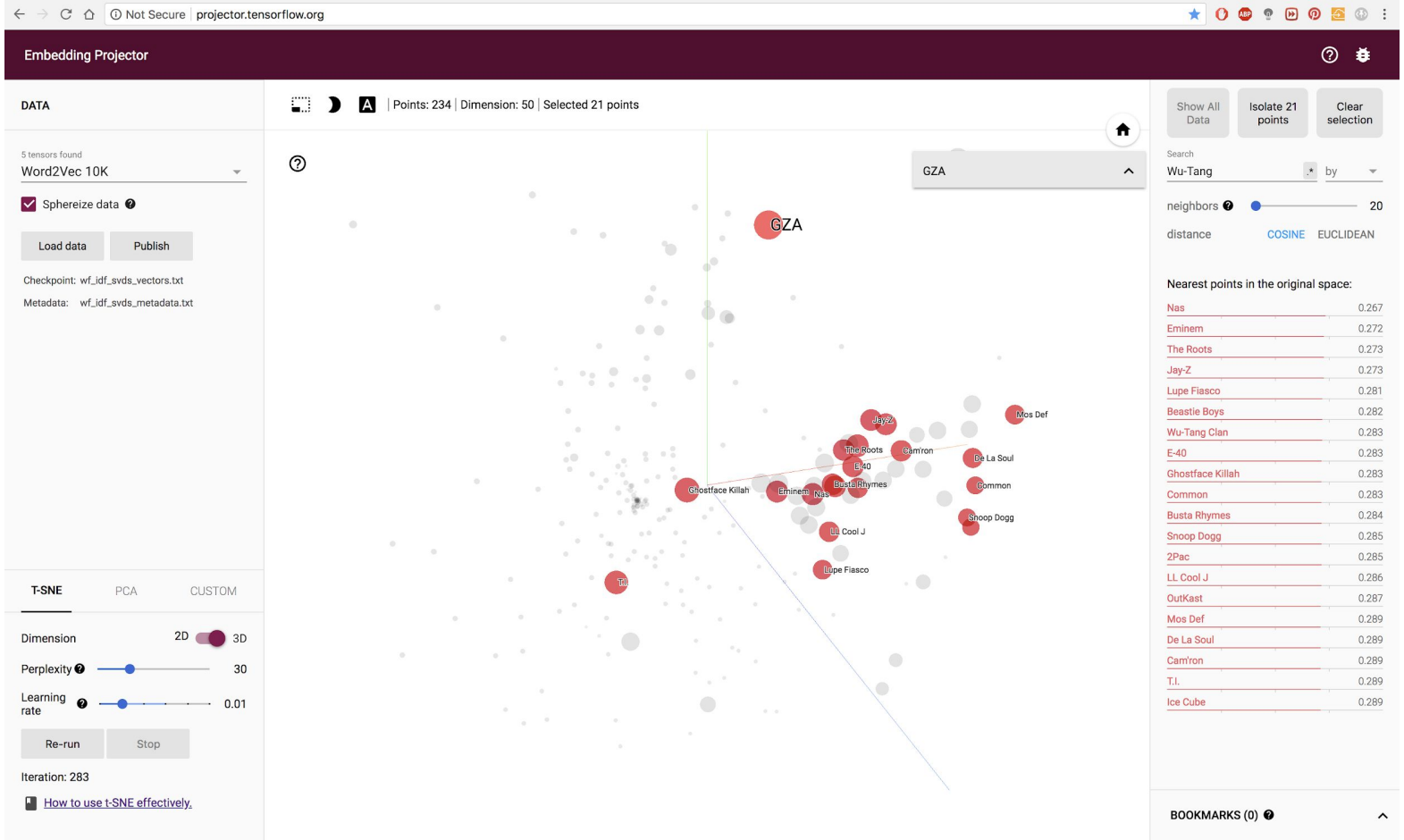
Final Graph - 2D // Epsilon (0.01), Steps (283), Perplexity (30)



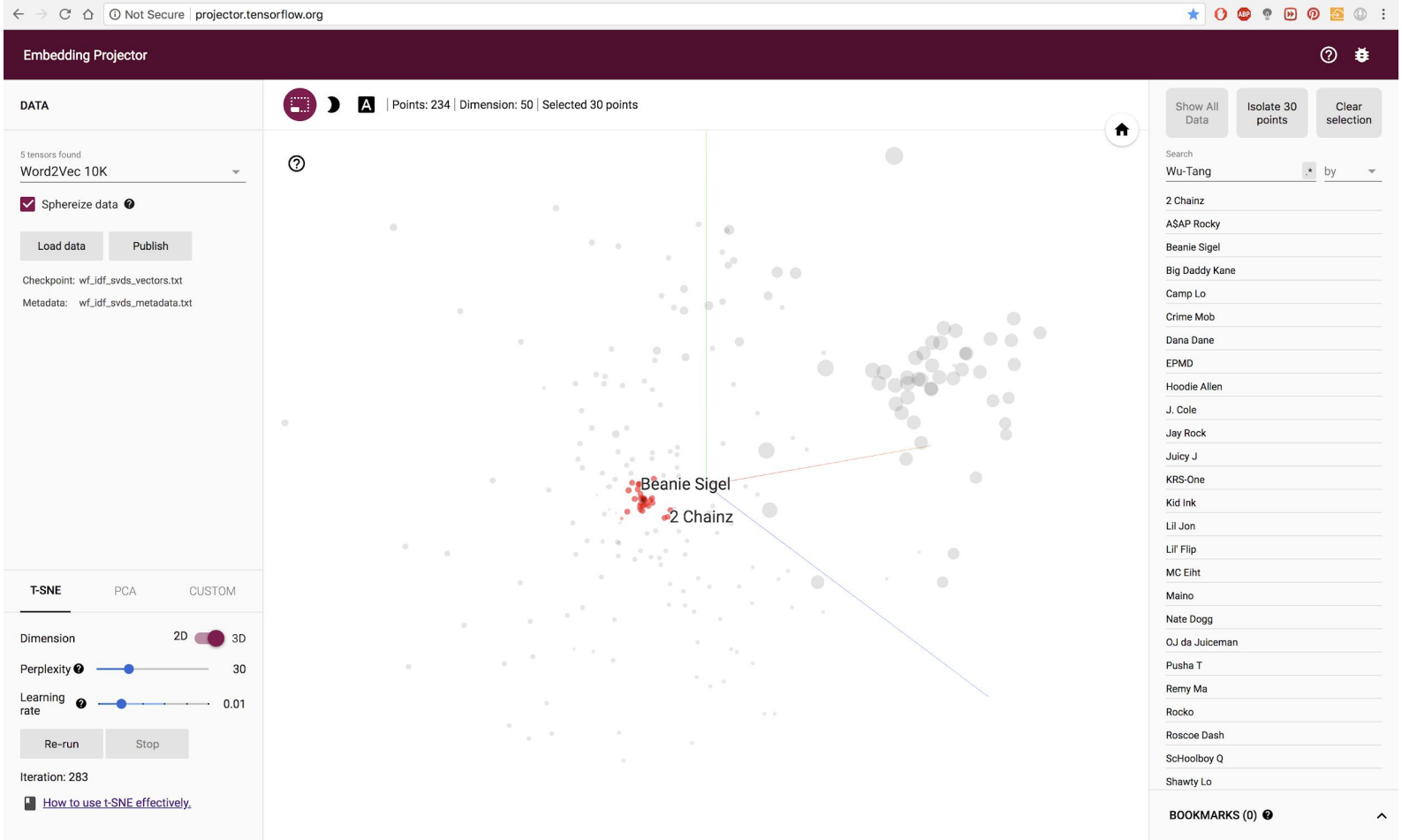
Final Graph - 2D // Epsilon (0.01), Steps (283), Perplexity (30)



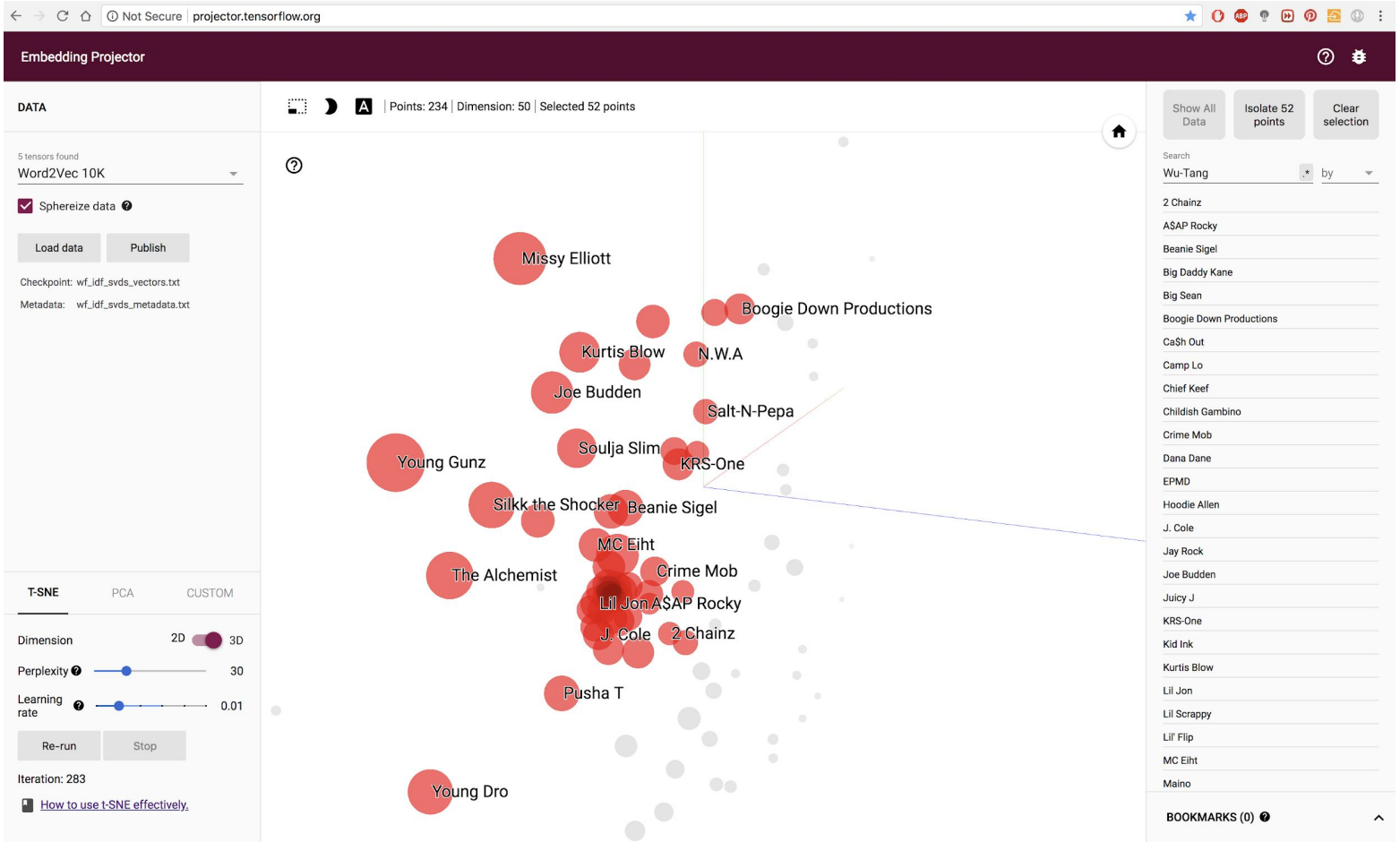
Final Graph - 3D // Epsilon (0.01), Steps (283), Perplexity (30)

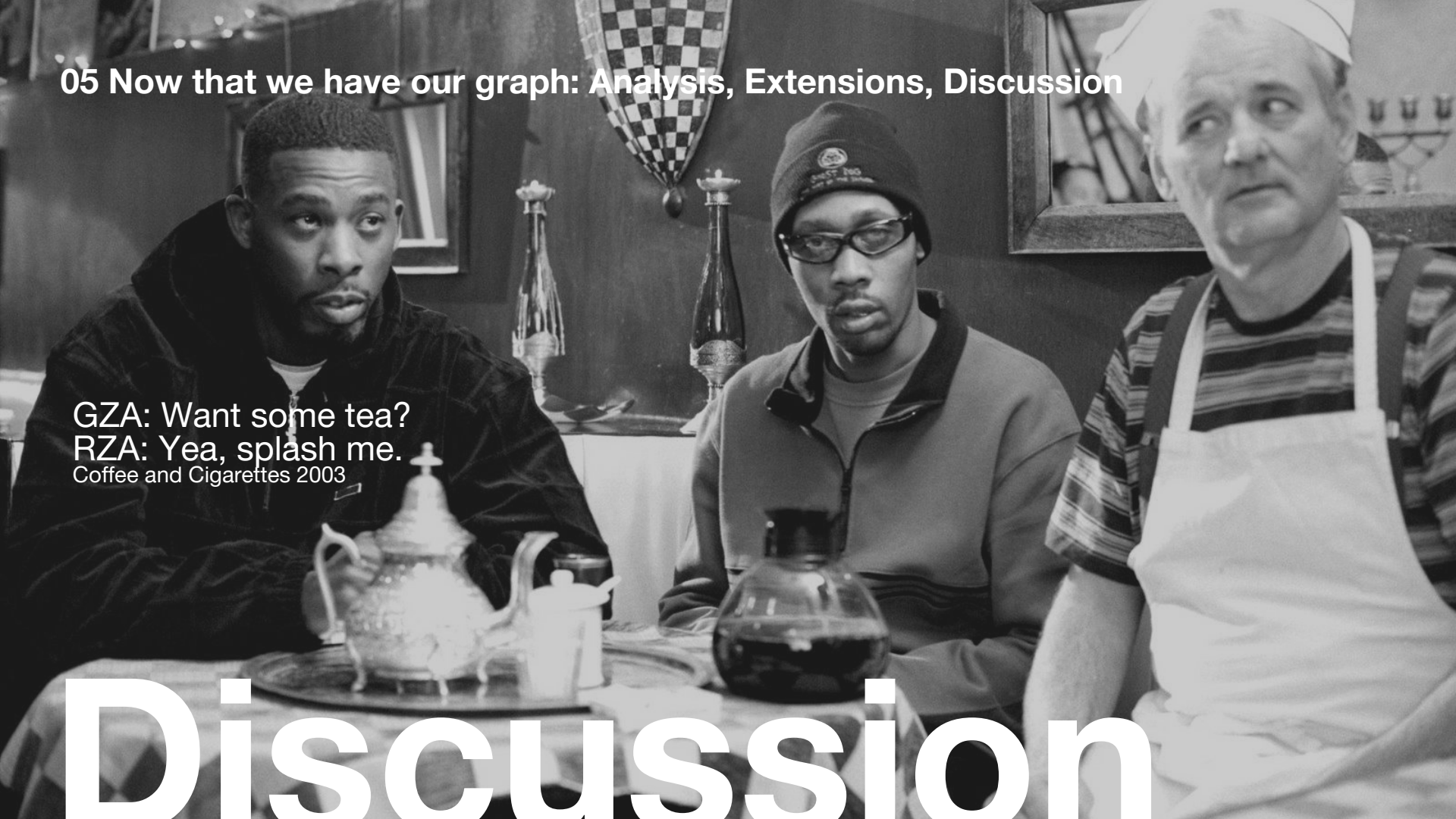


Final Graph - 3D // Epsilon (0.01), Steps (283), Perplexity (30)



Final Graph - 3D // Epsilon (0.01), Steps (283), Perplexity (30)





GZA: Want some tea?
RZA: Yea, splash me.
Coffee and Cigarettes 2003

Discussion

Graph theory concepts + Spotify algorithm mindset

- Compute artist graph for all genres, once a day, for the entire universe of artists
- Real-time recommendations computed by random walk from a given starting point (song or artist), the song or artist that comes up most frequently from a particular starting point in a random walk is the most relevant to the user - play this next
- Bias the walk toward user interests (other features), make the walks (relatively) short to keep relevancy high
- Prune the graph based on cosine similarity between the artist (song) and the rest of the artists (song) on the graph - increases relevancy and limits graph size to more usable real-time processing
- Think about how this works for Spotify's products: discover, discover weekly playlists, radio, related artists, etc...

Current Analysis Limitations

- Data I used vs. what *The Pudding* article used showed light differences
  - Article scraped their own hip hop data, used the LyricFind data to create the TF-IDF process
    - Missing some slang words in LyricFind data - lexical cleaning
  - Article used Truncated Singular Value Decomposition (T-SVD)
  - Tensorflow implementation most likely not through projector
- Good data + simple model > bad data + complicated model
  - Only feature I used was artist-lyric data, many other exist (playlist and user defined, song, use or lack of lyrics in a song, length or repetition of verses, featured artists on a song, etc.)

06 Conclusion - in where we are told what we have already been told

Wu Tang Clan ain’t nuthing ta fuck wit,
Wu Tang Clan ain’t nuthing ta fuck wit,
Wu Tang Clan ain’t nuthing ta fuck wit,
Wu Tang Clan ain’t nuthing ta fuck wit,
Wu Tang Clan ain’t nuthing ta fuck wit,
Wu Tang Clan ain’t nuthing ta fuck wit,
Wu Tang Clan ain’t nuthing ta fuck wit,

Wu-Tang Clan Ain't Nuttin ta Fuck Wit
Wu-Tang Clan - Enter the Wu Tang (36 Chambers) 1993
Universal Music Publishing Group

Conclusion

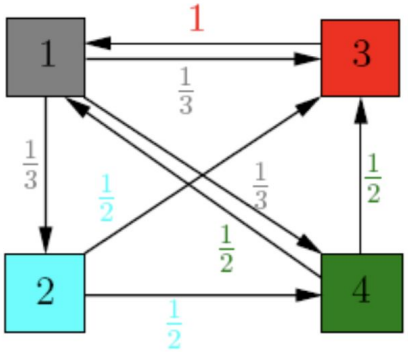
TL;DR

- Utilizing ideas from web structure, search algorithms, and graph theory, we analyze lyric data for a set of hip hop artists to find the language most endemic to the genre and most endemic to each artist in the genre through a technique called TF-IDF (term frequency - inverse document frequency)
- We find artists that are most like alike by computing the cosine similarity between artists' TF-IDF vectors
- We run into a dimensionality problem when computing similarity and creating a visualization, so we employ the Sparse Singular Value Decomposition algorithm (SVD-S) to reduce the language vector dimensionality
- To create our final graph, we employ a machine learning visualization algorithm - t-distributed stochastic neighbor embedding (t-SNE) - within a Tensorflow projector
- We then analyze the graph for meaning and discuss its possible use cases through the lens of *the Spotify algorithm*

Appendix

PageRank

- PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.
- In a web graph the nodes are the pages, directed edges are weighted by number of outlinks
- Steady-state probability distribution (dominant normalized eigenvector), v, is computed from the transition matrix or T, which represents the graph
- To account for random web surfing, PageRank algorithm adds a positive decimal damping factor q (usually 0.15) to the Transition matrix T
- Facebook graph for contrast is more deterministic than probabilistic



$$T = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \quad v = \begin{pmatrix} 0.39 \\ 0.13 \\ 0.29 \\ 0.19 \end{pmatrix} \quad G_{i,j} = (1-q)T_{i,j} + q/n$$