

Text Visualization of Song Lyrics

Jieun Oh

Center for Computer Research in Music and Acoustics
Stanford University
Stanford, CA 94305 USA
jieun5@ccrma.stanford.edu

ABSTRACT

We investigate techniques for visualizing song lyrics. Specifically, we present a design to tightly align musical and linguistic features of a song in a way that preserves the readability of lyrics, while conveying how the melodic, harmonic, and metric lines of the music fit with the text. We demonstrate and evaluate our technique on two examples that have been implemented using *Protovis*, and propose an approach to automating the data creation procedure using MIDI files with lyrics events as an input.

Author Keywords

Visualization, song, lyrics, text, music, focus and context, sparklines, saliency, tag-cloud

ACM Classification Keywords

H.5.2 Information Interfaces: User Interfaces

INTRODUCTION

Looking up song lyrics on the web is a commonly performed task for musicians and non-musicians alike. For instance, a user may type something like “Falling Slowly lyrics” in the search box to look up sites that show the lyrics to the song, “Falling Slowly.”

While the user can almost surely find the lyrics to songs of interest, the lyrics that are displayed as plaintext are not as useful by themselves; unless the user has available the audio file of the song to follow along, it is quite difficult to tell how the lyrics actually fit with the musical elements.

Thus, there is a strong need for a lightweight, easy-to-follow “musical guide” to accompany the lyrics. Ideally, the design should not detract from the readability of the text, while highlighting key features in the music. In contrast to having an entire musical score, which severely limits the target audience to those who can read western music notations (and furthermore require several pages worth of space to display), this visualization method would welcome a wider range of audience—including those who cannot read music—to convey musical features in a

space-efficient manner and provide a gist for an unfamiliar piece of music.

Though the original motivation for this visualization has been as described above, our design technique has many potentially useful applications:

(1) Research Data on Text Setting

Exploring the relationships between language and music has been an active area of research in the field of music perception and cognition. Musical scenarios that are particularly interesting to investigate are those in which the linguistic features of the text are in conflict with the musical features of the tune to which the text is set.

The source of conflict can be in pitch [3], accents [4], or duration [6] and the resolution of this conflict depends on the saliency of musical and linguistic features under contention: in some occasions singers will choose to follow the musical features at the cost of inaccurate linguistic production, while in other circumstances singers will distort the musical features to preserve the linguistic integrity.

Thus, having a text visualization of song lyrics would aid researchers hoping to have a quick look at how the text aligns with the music, in terms of accented beats, melodic contour, and harmonic progression.

(2) Automatic generation of “Song Cloud”

Similar to how general-purpose tag clouds [1, 5] describe the content of web sites, “song clouds” can be created to convey how a song is comprised of characteristic words in its lyrics, with their musical summary. Such a cloud can be generated almost immediately from our visualization design by extracting individual words based on structural organizations as well as musical- and linguistic-features that are salient. This possibility is further detailed under *Results*.

(3) Singing for Non-Musicians

Singing off of text alone by non-musicians is a phenomenon experienced quite frequently.

A prime example of this is during Church services: lyrics to hymns and other Contemporary Christian Music are often projected on a large screen for the congregation to sing from. In these situations, having an easy-to-understand musical summary presented along with the text would help not only the accompanists and instrumentalists who would appreciate chord-reminders, but also the majority non-musicians who would benefit from seeing the general pitch contour of the song melody.

A contrasting (but equally relevant) scenario is karaoke. Though most karaoke machines highlight the text as the words are being sung, it would be more informative to preview where the metric beats fall and how the melodic contour will change before it happens. Such could be made possible using a simplified version of our visualization.

RELATED WORK

The author is not aware of any existing work that visualizes song lyrics based on musical features. Two examples that are probably the closest to our goals are (1) **cantillation signs** (see Figure 1) and (2) **typography motion graphics** (see Figure 2). Interestingly, the former dates back to the medieval times to encode certain syntactic, phonetic, and musical information on the written text of the Hebrew Bible, while the latter has been a recent artistic trend to animate text over popular music.

In addition, our design incorporates concepts of a tag cloud [1, 5], song structure diagrams [8], and sparklines [7].



Figure 1. Cantillation signs. Letters in black, vowel points in red, and cantillation in blue. (<http://en.wikipedia.org/wiki/Cantillation>)



Figure 2. Typography Motion Graphics. Many examples of these art forms can be video searched under “motion typography” (http://en.wikipedia.org/wiki/Motion_graphic_design)



Figure 3. Tag Cloud Examples



Figure 4. Wattenberg's Shape of Songs

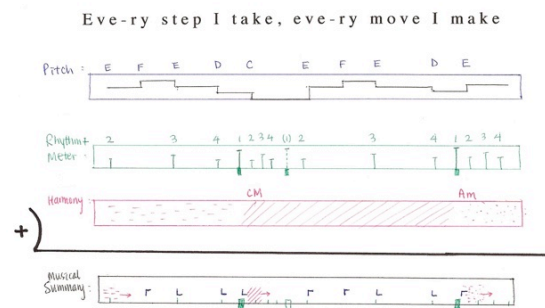


Figure 5. Prototype Sketch: Musical Sparklines

(3) Tag Clouds

A tag cloud (or word cloud, Figure 3) is a user-generated tag that characterizes webpage contents. Visual parameters, such as font type, size, and color, can encode the words' frequency or relevance in the content. Taking this idea, we have visually encoded several linguistic features of the song lyrics.

(4) Martin Wattenberg's Shape of Song

Wattenberg's *Shape of Song* (Figure 4) visualizes musical structure using diagrams. We incorporate a vertical timeline with labeled subsections to convey the structural organization of our songs.

(5) Edward Tufte's Sparklines

Finally, we incorporate Tufte's concept of a sparkline to encapsulate key musical features in a space equivalent to a line of text. Figure 5 shows a prototype sketch of how a “summary sparkline” can be calculated based on separate musical features.

METHODS

Approach: Three-Components Strategy

Our design is comprised of three major components: (1) sparklines that summarize musical features, (2) lyrics with chosen linguistic features encoded as bold and italicized text, and (3) a vertical timeline that shows the focused content of (1+2) in the context of overall song structure. These three components are tightly aligned, both horizontally and vertically, to maximally convey their relationships.

Implementation Details

Our design was implemented using *Protovis* [2], a graphical toolkit for visualization, which uses JavaScript and SVG for web-native visualizations. Examples of text visualization of song lyrics are best viewed using Firefox, though they are also compatible with Safari and Chrome.

Component 1: Musical Features

Three features that are consistently found across all major genres of popular music and are considered to be essential to the sound are melody, harmony, and rhythm. We have thus chosen these three features to encode in our musical sparkline.

Two functions, `sparkline` and `rapline`, encapsulate the rendering of each lines of sparklines. Specifically, `sparkline (first, last, rest)` is used to draw a sparkline for “pitched” music (or rests), and takes as parameters the starting and ending indices of the `letter` field in the data file, as well as a boolean value indicating whether we are rendering over rests. In contrast, `rapline (first, last)` is used to draw a sparkline for rap lyrics: it conserves the vertical space used for drawing the pitch contour in the `sparkline` function¹, and brings out the metric information by using thicker tick marks. We now describe the details of encoding melody, harmony, and rhythm using these functions.

Component 1.1: Melody

Melodic pitch contour is encoded as a line graph: `pv.Line` with `.interpolate (“step-after”)` is used to implement this feature, since a

¹ The rationale for conserving vertical space is to allow for easier horizontal alignment with the vertical timeline component. Because rap lyrics have high density, efforts must be taken to minimize the vertical spacing allotted to each line of text so that we do not need to unnecessarily increase the height normalization factor applied to the vertical timeline, to which the sparklines are horizontally aligned.

musical note is a discrete pitch that stays constant throughout its duration.

MIDI note number is used as the value to the `pitch` field in the data file, and this value determines the y-displacement (`.bottom`) of the line graph. The pitch contour maintains consistent height within a given line of text; absolute distance y-displacement should *not* be compared across different lines of text because the y-scale is normalized to the range (max – min) of pitch values for a given line of text. In doing so, we make the most use of the limited number of pixels available vertically to a sparkline.

Component 1.2: Harmony

Harmony is encoded as the color of the x-axis. The functional (i.e. Roman numeral notation) harmony is used as the value to the `chord` field in the data file, such that the 1=tonic, 2=supertonic, 3=mediant, 4=subdominant, 5=dominant, 6=submediant, 7=leading tone/ subtonic. For instance in a C Major key, `chord=5` would be assigned to a G Major chord (triad or 7th chord of various inversions and modifications), while `chord=6` would be assigned to an A minor chord (again, a triad or 7th chord of various inversions and modifications). Chords were encoded based on their functions rather than by the absolute pitch of their root to allow for transpositions.

`pv.Colors.category10` was used to assign colors to chord types. By using this built in category, we guarantee choice of perceptually easily distinguishable color values. To have a consistent chord-to-color mapping across songs, we can insert the following code in the initial lines of data (which do not actually get rendered, because our rendering functions, do not call on negative `letter` indices):

```
var data = [  
  {letter:-10, chord:1},  
  {letter:-9, chord:4},  
  {letter:-8, chord:6},  
  {letter:-7, chord:5}, ...];
```

Specifically, this code maps the first color (blue) in the `pv.Colors.category10` to the tonic chord, the second color (orange) to the subdominant chord, the third color (green) to the subdominant chord, and the fourth color (red) to the dominant chord, *regardless* of the order in which the chords first appear in the rendered portion of the data. Thus, we are able to make quick and direct comparisons of chord progressions across different songs by comparing color.

Component 1.3: Rhythm

Rhythm is encoded as the tick marks (`pv.Line`) along the x-axis. A longer tick-mark denotes the first



Figure 6. Component 1: Musical Features
 Consists of a step chart showing the relative melodic pitch contour of the phrase, a colored axis that encodes functional harmonic progression, and ticks marking the musical beats.

beat of each measure (i.e. downbeat), while a shorter tick-mark denotes all other beats. The *pulse* field in the data file encodes this metric information.

Figure 6 shows an example sparkline that combines all three musical features. In this figure, the melody jumps up then descends slowly, and harmonic chords progress from submediant (vi, Am) to dominant (V, G) to subdominant (IV, F) over two measures of 4/4. The final three beats are rests.

Component 2: Linguistic Features

A conscious effort was made to avoid any variable size- or spacing- distortions² in the display of lyrics because, despite the added musical information that our visualization offers, our design priority was in fulfilling the need of the primary use-case: lyrics look-up. Moreover, having a variable number of pixels per character would make it difficult to automatically determine the horizontal axis scaling for rendering the content of the sparklines (see subsection on *Alignment* below).

Also, because color was already used to encode harmony, we retained black as the text color, allowing for a visual separation between the musical sparklines and the lyrics.

As a result, simple `` and `<i>` tags were used to bold and italicize³ words based on their semantics and phonetics. Specifically, content words (i.e. words with high semantic values) were bolded, and accented syllables were italicized. Other linguistic features, such as rhymes or parts of speech, could have been chosen in place of these linguistic features.

² Loosening up on this requirement, we could consider altering the horizontal spacing of text to make the musical beats equally spaced out—such that horizontal layout can be perceived and interpreted directly as temporal duration. This method would be preferred if we prioritized musical performance over readability of text. It has been suggested that a checkbox that toggles between these two modes would be a useful feature in the design, and should be included to suit the needs of both use cases.

³ It turns out that bolding and italicizing does not change the width of a character of a monospaced font—an important requirement we must depend on to align the sparklines to the text.

you have *suffered* enough and *warred* with yourself.

Figure 7. Component 2: Linguistic Features
 Lexical and phrase accents are italicized and content words are bolded.

Component 3: Context & Overall Form

A vertical timeline of the song, displayed to the left of the contents of Components 1 and 2, is used to convey the overall form and musical structure.

The timeline (as shown in Figure 8) is broken into shorter pieces to denote the subsections of the song. The length of these bars preserve relative temporal duration, such that we can make accurate judgment on the proportion of time taken up by different sections in the music by comparing the height of bars. Each section is labeled with a time stamp and section name using anchored `pv.Label`.

Because popular music almost always starts (and ends) with an instrumental section that serves as a bookend to the main parts with vocals, a lighter gray color was used to shade these instrumental “intro” and “outro”. As a result, one can more easily compare their lengths to the rest of the song.

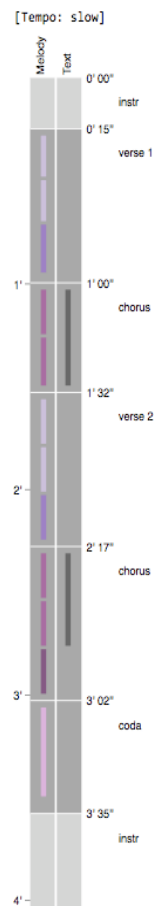


Figure 8.
Component 3:
Context & Overall Form

Vertical timeline shows major sections of the song with labels and timestamps. Tempo category is displayed at the top. Color encodings convey similarities and repetitions in the melody (left column) and in the text (right column).

To further elaborate on the vertical scaling of the timeline, the factor by which the height of `pv.Bars` is adjusted is determined by the tempo category of the piece. For instance `tempo=1` is used for relatively slow songs (such as *Falling Slowly*) with sparse text-per-beat ratio, while `tempo=2` is used for faster songs (such as *I'll be Missing You*) with higher text-per-beat density. Thus, a song categorized as `tempo=1` that lasts for 3 minutes will actually take up half the vertical length of a song of same duration categorized as `tempo=2`. Tempo categories had to be created to make offer more display-space to songs with high words density, while minimizing unused space for songs with low words density. Consequently, one can still make a direct comparison in duration between songs within the same tempo-category by the absolute-length of the bars in the timeline.

Finally, color encodings convey similarities and repetitions in the melody (left column) and in the text (right column). For instance, Figure 8 shows that the same melody is used in the first two-thirds of verse 1 and verse 2, and all of the text in the first chorus section gets repeated in the second chorus section. Though the colors used in current implementation are hard-coded as a categorical variable, it is not difficult to envision making an ordinal encoding of color based on similarity score used to compare a string of pitch-values (to determine melodic repetitions) and letters (to determine text repetitions).

Alignment of Components

Enforcing a tight alignment of the three components was at the crux of our visualization design.

We are able to *vertically align* the sparklines with the lyrics by using a monospace font for displaying the lyrics. Specifically, we have chosen 12pt Consolas font, which has width of 8.8 pixels per character. In this way, the character index in the data file determines the horizontal displacement of musical features to be rendered.

As shown in Figure 9, tick marks are drawn right above the first vowel of the syllable on which the musical beat falls. This is consistent with how the text is aligned to music in a traditional music score. As for the line graph, transitions in pitch values are aligned to their exact places in the text, which usually occur at word boundaries. Harmonic changes occur right on the musical beat.

Falling slowly, eyes that know me and I can't go back.

Figure 9. Vertical Alignment:
The musical sparklines and lyrics are vertically aligned using a monospaced font.

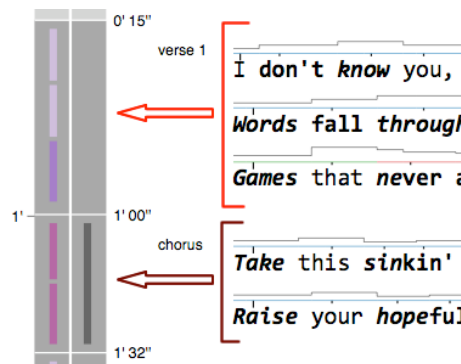


Figure 10. Horizontal Alignment:
The content to the right are horizontally aligned to the structural timeline to the left.

We also enforce a *horizontal alignment* between the focused content (component 1+2) and the context timeline (component 3). To do so, we specify line-height of the text by pixel number to override any browser-dependent default values. For a similar reason, we create our own custom breaks and spacers based on hard-coded pixel values.

RESULTS

In this section, we demonstrate and evaluate our technique that have been tested on two song examples: *Falling Slowly* from the *Once* soundtrack by Glen Hansard and Marketa Irglova, and the *I'll be Missing You* featuring Faith Evans and 112 by Puff Daddy. Figure 11 shows a final visualization of *Falling Slowly*.

Because the size of the JavaScript data file associated with a song is relatively large, it takes about 3-8 seconds for the browser to connect to source and render the visualization.

Also, current implementation involves creating the data file and tweaking the alignments manually, which is tedious and time consuming; in Section 6 we propose an approach to automate much of this process using a symbolic input data.



Figure 11. *Falling Slowly* Final visualization created by combining the three major components

Evaluation

Pros

Our visualization features a compact yet lossless encoding of relative pitch contours, harmonic functions, and metric beats using the concept of sparklines. In contrast to having a full music score that takes up several pages, or the other extreme of having no musical information provided at all, providing a musical summary offers the essential information to help the user understand how the text fits with music. Furthermore, having a tight alignment across components place the focused content in the context of the overall song structure. All of this is achieved without distorting the size or spacing of the lyrics, thus maintaining the level of readability comparable to that of a plain text.

Cons

Because we have attempted to create a compact summary of the musical features, the resulting visualization may be criticized for having “too much ink”; the density of information could potentially

overwhelm the user. There may be a creative way to minimize ink used in the sparklines, such as by getting rid of the x-axis link altogether, and showing the harmonic color and metric ticks directly on the line graphs. However, this is likely to lead to other problems, such as difficulty in telling the relative height of the line graph (if we remove x-axis that currently serves as the reference line) as well as confusing the tick marks from the vertical transition jumps in the line graph itself.

Also, current design does not have the flexibilities to encode polyphonic melodies, such as duets (as is the case for certain sections of *Falling Slowly*).

Other limitations include lack of encoding schemes for including subtle expressions and nuances, such as dynamics and articulations, which are, after all, key element that make music beautiful and enjoyable.

Prototype Song Cloud Example

One of the applications of our visualization is an automatic generation of song clouds. By extracting

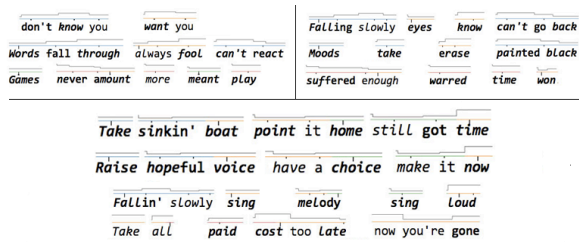


Figure 12. Song Cloud Example: *Falling Slowly*
 Cloud shows structure (verse 1 and 2 at top left and right, chorus in the bottom) and frequency (encoded as size) of salient words in lyrics.

individual words based on structural repetition or salient musical- and linguistic-features, we can put together a characteristic cloud that captures musical and linguistic features of a given song.

Figure 12 shows a prototype example of a song cloud created directly from our *Falling Slowly* visualization.

DISCUSSION

An informal observation of user reactions of the visualization suggests that the encodings of parameters in the musical sparklines are not as self-explanatory as we have hoped. That is, users had to be explained how the melody, harmony, and metric beats are encoded in the sparklines. The tick marks that indicate musical beats, in particular, seemed counter-intuitive to many, and the non-musician users who are unfamiliar with music chords did not find the color encodings of the axis informative, beyond seeing patterns of change.

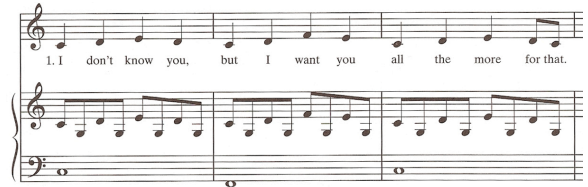
At the same time, users have commented that the visualization reminds them of karaoke and games such as *Guitar Hero*, which suggests similarities with these systems in ways that pitch and rhythmic information are encoded. However, one should remember that in contrast to animated games such as *Guitar Hero*, our visualization is static and thus overcomes the challenge of making the most out of limited screen space.

Interestingly, users who were at least somewhat familiar with the song examples were more motivated to follow along the musical summary to check with their knowledge of the song. In this manner, the visualization was found to be better suited to those who are interested in finding out more about the musical features of songs they know moderately well (or conversely, want to check the precise words set to the tune that they are familiar with).

The author was able to make some interesting observations herself while creating the visualizations for the two example songs—suggesting the visualization’s potential to uncover interesting patterns that are otherwise difficult to see and hear. For instance, the first sentence in verse 1 of *Falling Slowly* is typically written out as three short lines:

*I don't know you
 But I want you
 All the more for that*

The music set to this text is also segmented into three measures with similar melodic contours and rhythm:



Thus, it is difficult to hear the meaning of the sentence in its entirety and parse it correctly as two unequal chunks:

*I don't know you,
 But I want you all the more for that.*

In the process of creating the visualization, which carefully inserts line breaks only at sentence endings, the author for the first time realized the correct semantic parsing of the sentence: that “all the more” modifies the verb “want”, while the pronoun “that” refers to “not knowing.”

The visualization also helps with ambiguities that arise in determining whether a syllable falls *on* the downbeat or *prior to* the downbeat as an anacrusis. A prime example is in line 2 of *Falling Slowly*:

Words fall through me and always fool me and I can't react.

Note that the first “and” is a pick-up note to the downbeat of the second measure, while the second “and” actually falls right on the downbeat of the third measure. Given only the text, one must guess how to place the word “and” in each phrase, but the ambiguity clears up with the presence of tick marks shown on the sparkline.

Finally, the visualization encourages exploration on the interplay between linguistic features and musical features. In *I'll be Missing You*, for example, one may look for instances in which lexical and phrase accents are “misaligned” with the metric beats. Interestingly, we find that many multisyllabic words are intentionally *misaligned* to the metric beats, and it is perhaps this type of idiosyncrasies that make rap beats interesting to listen to. Figure 13 shows examples of such misaligned multisyllabic words.

yesterday future stages family
yesterday future stages family

Figure 13. Conflicting Lexical and Metric Accents
Examples in *I'll be Missing You* in which the syllable on which the metric beat falls is different from the syllable that receives lexical accents

FUTURE WORK: AUTOMATING DATA CREATION

As previously mentioned, there is a strong need to automate the data file creation process, for it is very tedious and time-consuming to do by hand.

As for automatically generating the data for sparklines, a MIDI file with lyrics events should provide all the necessary input data needed to create the sparklines (though chords may have to be calculated separately based on the individual notes). Alternatively, one could consider using Music OCR to scan sheet music and convert the information to a format in the analytical domain. In either case, the challenge lies in formatting the data appropriately so that it can be understood by Protovis. For a given line of song (text + music), the data used by Protovis should contain all instances of character indices (in an ascending order) where (1) the pitch changes, (2) the metric beat falls, and (3) harmonic chord changes. As long as we know how many letters are in a given word, how the word is divided into syllables, and where the first vowel of each syllable lies, the index positions can be determined.

As for formatting the lyrics according to phonetics and semantics, we would need to access a simple dictionary that divides the words into syllables, shows the accented syllable, and gives an approximate idea as to the word's semantic saliency.

Finally, an algorithm should be written to compute the similarities in the melodic contour and lyrics between lines of music. This would then be used to encode repetition in melody and text in the vertical timeline, and the similarity score could then be mapped to a color using ordinal scale.

Once these procedures are automated, there would be only a small amount of work remaining for the human to make final touches on the html file. Processes that are likely to require some human intervention are (1) segmenting lyrics into lines (such as on every sentence breaks) in a way that preserves the semantics of text as much as possible, (2) determining the tempo category for the song based on the text density, and (3) inserting spacers between

sections of music to horizontally align with the timeline. But provided that the size of all margins and fonts are specified in absolute pixel amount, even these remaining steps could eventually all be automated.

ACKNOWLEDGMENT

The author wishes to thank Professor Jeffrey Heer, Mike Bostock, and rest of class CS448b for their feedback and guidance.

REFERENCES

1. Bateman, S., Gutwin, C., and Nacenta, M. Seeing things in the clouds: the effect of visual features on tag cloud selections. In *Proc. Hypertext and Hypermedia 2008*. ACM Press (2008), 193-202.
2. Michael Bostock, Jeffrey Heer, "Protovis: A Graphical Toolkit for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1121-1128, November/December, 2009.
3. Mang, E. (2007). Speech-song interface of Chinese speakers. *Music Education Research*, 9(1), 49-64.
4. Morgan, T. & Janda, R. (1989). Musically-conditioned stress shift in Spanish revisited: empirical verification and nonlinear analysis. In Carl Kirschner, and Janet Ann DeCesaris (eds.) *Studies in Romance Linguistics*, Selected Proceedings from the XVII Linguistic Symposium on Romance Languages. New Jersey: Rutgers University.
5. Rivadeneira, A.W., Gruen, D.M., Muller, M.J., and Millen, D.R. Getting our head in the clouds: toward evaluation studies of tagclouds. *Proc. CHI '07*. 995-998.
6. Ross, J. (2003). Same words performed spoken and sung: An acoustic comparison. *Proceedings of the 5th Triennial ESCOM Conference*, Hanover University, Germany
7. E. Tufte, "Sparklines: Theory and Practice." http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001OR.
8. Wattenberg, M., "Arc diagrams: visualizing structure in strings," *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, vol., no., pp. 110-116, 2002

APPENDIX

All visualization examples, code, and other files can be found at:

<https://ccrma.stanford.edu/~jjeun5/cs448b/final/>