TRIQTUNES

FRANK GOODMAN, KEVIN PAETH, ANKIT SHAH, ARUN VARMA

SONG SUGGESTIONS FOR LITERATURE ON THE WEB

ABSTRACT

Browsing the web can be (and is increasingly) enhanced by the accompaniment of music. However, the process of determining appropriate song selections can be a time sink, and a poor choice can be detrimental to focus and overall experience. Thus, Triqtunes is a Google Chrome extension aiming to address and alleviate these issues by automating the process. Namely, songs are chosen based on current browser content and played automatically.

METHODOLOGY

PREPROCESSING

We use musiXmatch dataset (official lyrics collection of Million Song Dataset) for word frequency. This provides a bag-of-words model for lyrics of 779,000 songs, and adheres to copyright laws. We augment this with song ratings from Spotify API and sentiment scores from AFINN. We construct a feature space characterized by word frequency, popularity, and song sentiment to determine song selection.

TEXT SELECTION

Users with the Triqtunes Chrome extension select text in-browser, which sends a GET request with the highlighted text to our Flash server on Amazon AWS.

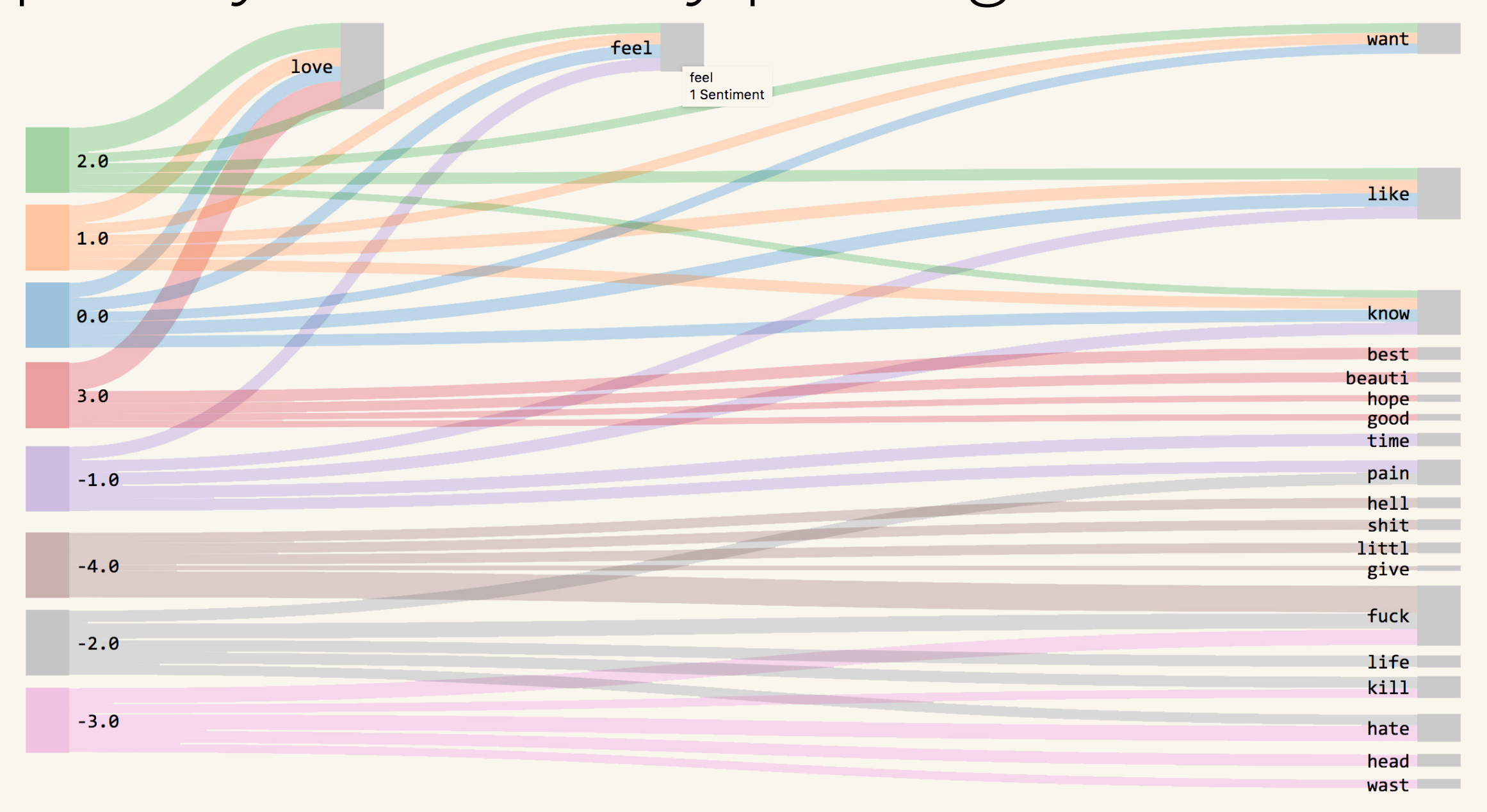
RECOMMENDATION ENGINE

Our first suggestion pipeline focused on word frequencies of the article's strongest words, and searched through our database for songs indexed by these words as lyrics. The engine returned songs that share similar content. This worked with decent success, but was erratic in its results.
Eventually, we featurized data by word frequency and sentiment. Our next iteration generates suggestions based on performing a K-Nearest Neighbors algorithm on this feature space. Suggestions were more diverse, and their relevance more apparent in face of the article. They were also more likely to match the article's sentiment.

VISUALIZATIONS

SANKEY DIAGRAM: SONG SENTIMENTS

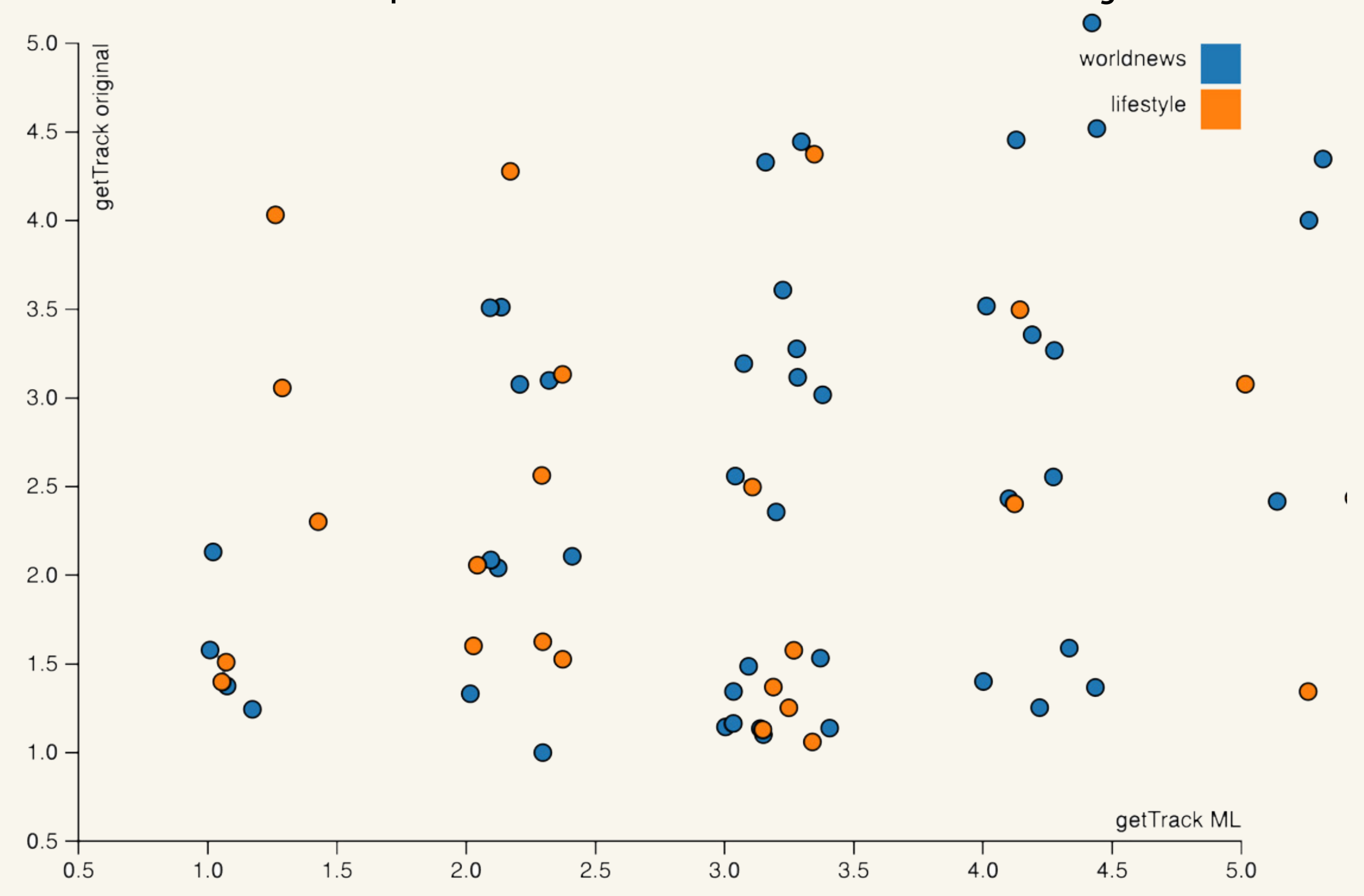
- Concept: map song sentiments to most popular associated words
- Sentiment values from AFINN
- Only include non-trivial words (non-stopword, length four or greater)
- Store songs in corresponding interval (i.e. songs with sentiment 2.3 maps to the interval 2-3)
- Map intervals of sentiment to most popular words associated with songs in this interval
- Excluding non-English songs by verifying most lyrics in AFINN
- Dump result as JSON object to interpret by D3's Sankey package



Not surprisingly, the most positive word is *love*, and the most negative word is *fuck*.

SCATTER PLOT: ENGINE COMPARISON

- Concept: visualize crowdsourced ratings of top suggestions from both KNN and no-ML suggestion engines
- Blind rating between 1 and 5; 1 signifies irrelevance and 5 full relevance
- Based off 70+ different articles scraped from *The Guardian*'s World News and Lifestyle sections
- Data dumped as CSV file for D3.js



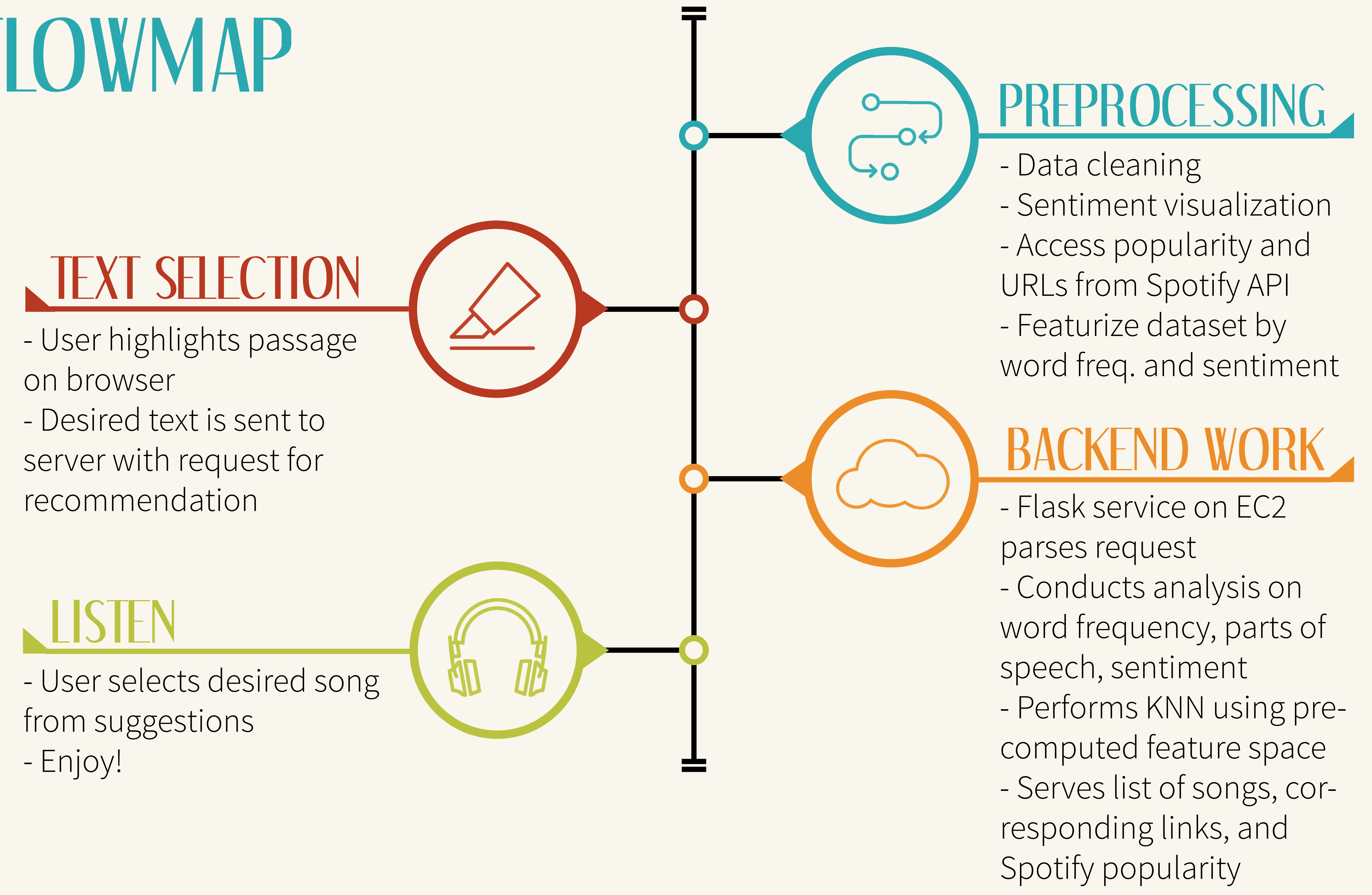
Data Averages

KNN, World News: **3.01**; No-ML, World News: **2.22**
KNN, Lifestyle: **2.80**; No-ML, Lifestyle: **2.19**
KNN, overall: **2.94**; No-ML, overall: **2.21**

RESULTS

As a proof of concept, we hoped to simply find songs with some relevance to given text, and the most direct way to compute this was simply matching popular words in our article to songs that contained them. Our success with this method was varied, with some relevant suggestions but also some noise. Suggestions provided by our naive method were too limiting. By comparing bag-of-words sets in a larger feature space using K-Nearest Neighbors across all words in all songs in our data set, we generalized suggestions better. On top of 4500 word dimensions, emphasizing a dimension for text sentiment meant suggestions were likely to match the article's mood. *Note*: our application is not very emotionally sensitive, and is less useful for browsing news dealing with somber and aggressive news stories. For example, most suggestions to articles on weather and first-person stories are appropriate in sentiment. Suggestions to articles on armed conflict in Syria were reliably uncomfortable. (See: TayTweets by Microsoft)

FLOWMAP



PREPROCESSING

- Data cleaning
- Sentiment visualization
- Access popularity and URLs from Spotify API
- Featurize dataset by word freq. and sentiment

TEXT SELECTION

- User highlights passage on browser
- Desired text is sent to server with request for recommendation

BACKEND WORK

- Flask service on EC2 parses request
- Conducts analysis on word frequency, parts of speech, sentiment
- Performs KNN using precomputed feature space
- Serves list of songs, corresponding links, and Spotify popularity

LISTEN

- User selects desired song from suggestions
- Enjoy!