University of Washington

HCDE – Visualizing Topic Models & Jokes

Visualizing the results from LDA topic modeling on the reddit joke dataset

Mohammed Helal, Vivek Pagadala, Luke Waninger
12-11-2017

Contents

Abstract

Text data is one of the most abundant sources of data available today, yet it is one of the most difficult sources to understand and gain insight from. We chose to tackle this problem in a tag-team approach of using machine learning and effective data visualization techniques to help us understand the topics of a dataset. We used a dataset scraped from Reddit's r/jokes subreddit and implemented a topic modelling algorithm to derive joke topics, and then implemented a visualization in d3 to maximize the ability of a user to understand these topics.

Final visualization: http://lkwaninger.ddns.net/

Introduction

The Reddit Joke Dataset

The dataset used had a format displayed below.


```
"title": "My boss said to me, \"you're the worst train driver ever. How many have you derailed this year?\
"body": "I said, \"I'm not sure; it's hard to keep track.\"",
"id": "5tyytx",
"score": 3
```


Each row in the dataset contained a joke title – which is often part of the joke – the joke body, an identifier, and a score. The score signifies the number of upvotes the joke received.

The conventional method of doing so involves implementing a topic modelling algorithm, and then reviewing the results to explore the topic outputs. The naming convention of topic modelling algorithms are arbitrary, so it is up to the user to identify which topics were attained by the algorithm. The purpose of this visualization is to explore the results to identify which topics were discovered.

The entire dataset included approximately 200 thousand jokes.

Topic Models

Here we introduce the topic modelling problem and the algorithm used to create the dataset used in the visualization.

A topic model is a statistical model for discovering abstract topics that occur in a collection of documents. It is frequently used in text-mining for discovery of hidden, or *latent*, semantic

structures in text. All topic models used today are based on the co-occurrence of words in a document. Intuitively, words that are used together often are likely to be associated.

Topic models can be thought of as clustering documents based on the words used in each document. However, a topic model extends this one step further, by allowing documents to be composed from a distribution of topics. This is desirable, since most documents are not simply about one topic. For example, a single magazine may discuss fashion, accessories, clothing, and fragrances all in one magazine. There may or may not be a hierarchy in the topics, but nonetheless, there should be a means of having multiple assignments to a document, and topic models allow for just that.

Latent Dirichlet Allocation

We will start by discussing the implementation of the algorithm used to generate the data for our visualization. We will not dive into the specifics of the algorithm as it is outside of the scope of this paper, but we will attempt to familiarize the reader with the intuition behind the method. The algorithm chosen is a popular method used for topic modelling. Latent Dirichlet Allocation (LDA) was developed by Blei et al. in 2002, and stems around the following assumptions of the document generation method.

For each document:

- Choose the number of words, N.
- Choose a mixture of topics. E.g. 20% topic 1, 30% topic 2, 50% topic 3.
- Generate words in a document by:
    - Picking a topic based on the documents multinomial distribution (in the 2nd step).
    - Picking a word based on the topic-word distribution.

The last bullet implies that each word has a probability of being sampled from a given topic. Therefore, the document generation method assumes that there is (1) a distribution of topics for each document, and (2) a distribution of words for each topic.

Documents are obviously not generated in this fashion, but these assumptions greatly simplify the problem as it provides a basis for the algorithm to infer topics based on co-occurrences of words in a joke.

The algorithm starts by randomly assigning words to topics with equal probabilities. These probabilities are then corrected at each iteration, along with the distribution of topics for each document.

The output of the LDA algorithm is a probability distribution of words to topics. For example, the word ‘man’ may have a probability of 0.1 of being associated with topic 1, 0.2 with topic 2, and so on until topic K – the final topic.

Notice how the LDA algorithm assumes we know the number of topics beforehand. One way of choosing the number of topics is to fit the algorithm on a portion of the documents (say 80%),

and then compute the *perplexity* on the rest (20%). The perplexity can be interpreted as a measure of uncertainty in our topic assignments. One way to reason about it is to ask what would it mean if the probability distribution of a word to topics was uncertain. Surely, it would mean that we don't have a clear idea of which topics are actually associated with this word, so for a given word we would expect to see equal probability to each topic. This would signal that the algorithm has no idea how to actually assign the word. Indeed, the perplexity is highest when the probability distribution of all words are uniform – meaning each topic has equal probability. As we become more certain, the distribution takes shape that skews one way or another, and the perplexity will decrease.

The aforementioned hold-out method (80%/20%) was done using several candidate number of topics, and the perplexity was computed each time. In reality, however, we used a procedure called 5-fold cross-validation. We did not want to arbitrarily select the 80%/20% split, so we partitioned the data into 5 chunks randomly, and fit the model on 4 of the chunks, and computed the perplexity on the last chunk, and then repeated for every combination so that every chunk was in the hold-out set once. This ensures there is no bias in our method of sampling. The result of this method is shown below.

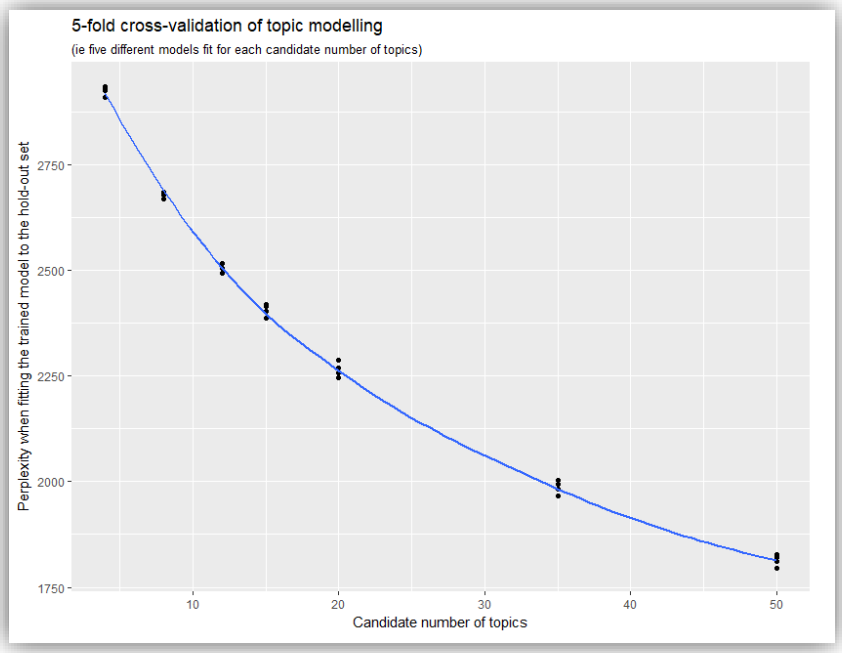

Figure 1 5-fold cross-validation scheme

We stopped the candidate topics at 50 for computational reasons. It seems that the perplexity is minimized here, so we chose the optimal number of topics to be 50.

Data Preprocessing

The joke data needed some preprocessing before passing it to the algorithm. The word “Want” and ‘want’ should be identified as the same, so the first step is to convert everything into lowercase characters. Similarly, punctuation was removed, so words like “wasn’t” were converted to words like “wasnt". Finally, some words are used often but are not very helpful in determining topics, such as “the”, “at”, “for”. These are called stop-words, and were removed during the preprocessing phase.

The joke title and body were then merged into a single document. Some tasks such as removing new lines and anything that wasn’t alphanumeric were also performed. For a detailed implementation, please refer to the link in the appendix for the github repository.
The final set of jokes consisted of approximately 28 thousand words.

Model Fitting

The actual input to the LDA algorithm must be a document-term matrix. An example of one is shown below.

Table 1 Example document term matrix

	man	woman	king	queen	…
1	1	1	1	1	…
2	1	0	1	0	…
3	0	1	1	0	…
…	…	…	…	…	…

The rows are the jokes, with the left-hand column being the joke number. The columns are the words in the corpus. If a joke contains a word, a 1 will be placed in the matrix, and 0 otherwise. This is referred to as a ‘bag-of-words’ model, since we are not considering the relative placement of the words with respect to one another, but rather whether they were used in the same joke together. This is an obvious flaw in the LDA algorithm, but there is yet to be an effective topic modelling algorithm that can capture semantic context in that fashion.

We finally trained the LDA algorithm on the entire corpus using 50 topics.

Now that the model is fit, we are given two outputs:

1. A matrix of word-topic probabilities. For each word-topic combination we have a probability indicating the probability of the word being associated with that topic.
2. A matrix of joke-topic probabilities. For each joke-topic combination we have a probability indicating the probability of the joke being associated with each topic.

The latter is simply formed from averaging the word-topic probabilities, and normalizing so that the probabilities add up to one.

We can use the matrix of word-topic probabilities to identify which topics are associated with which topics. We can do the same for jokes, but here we can also assign a joke a topic by taking assigning it the topic with the highest probability.

We are finally ready to start exploring the topic results in a visual setting. The following sections will outline the methodology and visualization techniques used to design and implement the final results.

Methodology

The approach we took to tackle this problem was inspired by Ben Fry's visualization process and the Five Design-Sheet methods. Here we will summarize the various tasks necessary to complete this project and the iterative nature of the design process that led to the final result.

We started this project with a question already in mind – that is, how can we best visualize the results of a topic model on the joke dataset. This question narrowed down our focus on the user experience with exploration in mind. However, we still did not know who the user would be, what was already out there, what kind of data-related tasks were required, or what type of visuals and interactions effectively achieved this goal. To answer these questions, we ended up with a process that looked like the plot below.

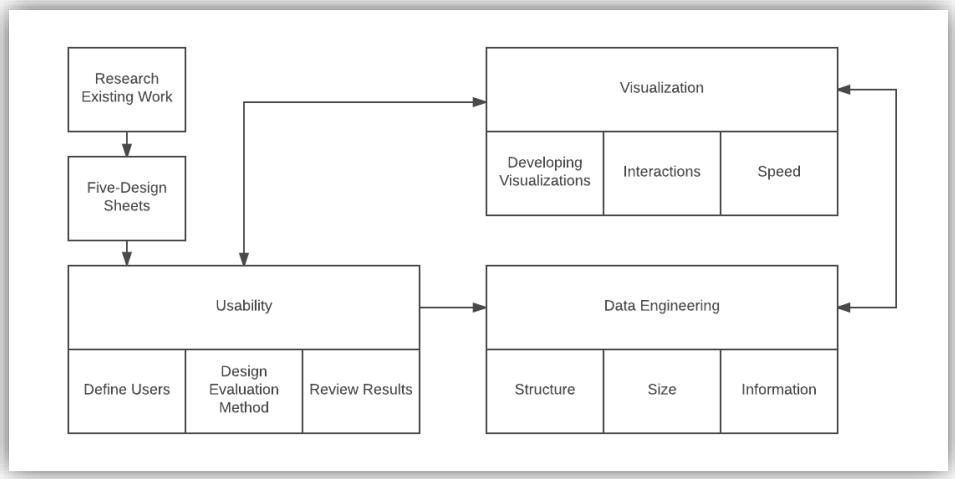

Figure 2 Design methodology

We started the process with researching existing visualization techniques used to visualize topic model results and identified flaws and disadvantages of each. We then used the Five-Design Sheets method to develop our drafts of the visualizations and the related functionality.

The next step was to get a better grasp of how the visualization would be used. We defined our users, and developed an effective method of evaluating the use cases we prioritized. Finally, we reviewed the results and adjusted our mockups accordingly.

Following that we were ready to begin the implementation process which alternated between data engineering and implementing the visualizations. The data engineering process required structuring the data in a usable format for the visualizations, as well as ensuring a structure that minimized lags in the visual interaction process. Consequently, the data also had to by filtered to keep the size manageable, so questions such as what is the best way to filter the data while not limiting the effectiveness of the visualization came to play. The designs were iterated, and occasionally new information was needed for the visualizations, and the data had to be brought into the pipeline.

The 'final' block is implementing the visualization in d3. This involved creating the bones of the visualizations, which were usually static, developing interactions, and ensuring the interactions were fast enough to not hinder the experience.

Notice how the visualization process feeds back and forth with usability. Although we only conducted a single usability study, we did conduct our own analyses using their feedback in mind, as well as asking friends about their opinions. Many iterations were made before coming up with the final design.

Research

The first step was searching for visualizations designed to explore topic model results.
The most popular method of visualizing LDA results is using an R package called LDAviz. A snapshot of what the results look like are shown below.

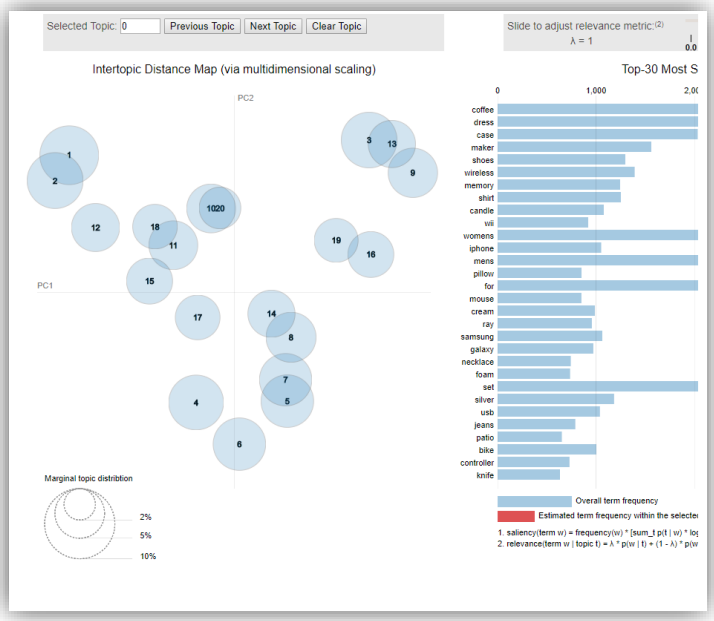

Figure 3 Source: https://www.kaggle.com/solution/lda-visualization

Although the visualization is aesthetically pleasing, the "intertopic distance" may be a bit confusing to those unfamiliar with dimensionality reduction. What the visualization does do well is use the barplot of words to effectively display the association of topics with words.

Another example is using the work of a prolific data scientist at StackOverflow, Julia Silge. In her blog, she fits a topic model to several books to test whether the models accurately group the pages into the correct book. The plot below is used on her blog when reviewing the results.

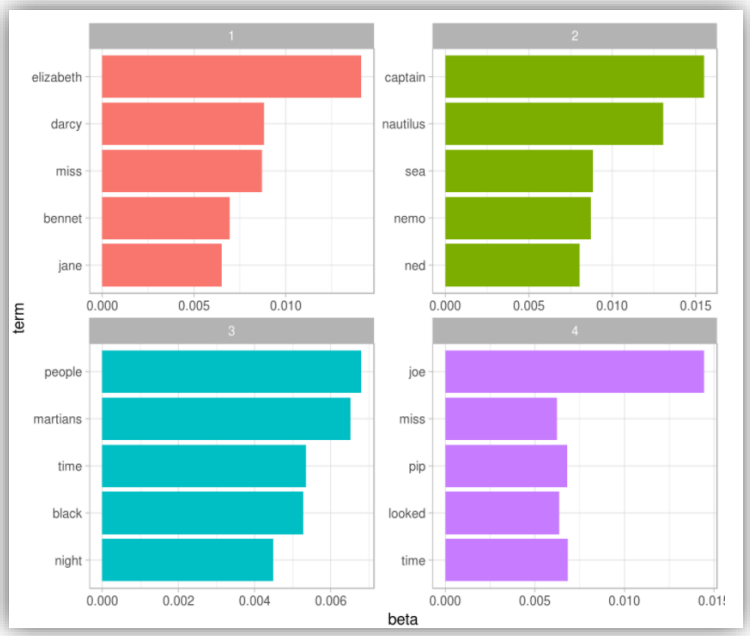

Figure 4 Source: http://tidytextmining.com/topicmodeling.html

We noticed that most attempts at inferring the results of a topic model involve barplots, which make sense – we are comparing the topic words (categorical) and encoding a measure of association using the length. It is an effective means of displaying the relationships.

However, what the plots lack is the ability to reference the text directly given these words. What if the set of words were not enough to identify a trend and wanted to look at the text that was assigned to this topic? Moreover, what if words are used frequently between two different topics? The plots make no way to show how similar the topics are. Finally, we have an ancillary goal; that is to identify which topics tend to get rated higher.

Overall, we were pleased with the current results, but certainly felt there were more opportunities we wanted to pursue in our own visualization.

Design Process

Five-Design Sheets

The creation of the Five-Design Sheets was to conceptualize and develop the first draft of our final project visualization.

The first sheet – the ideation phase – was initially done individually by each team member. Each of the team members came up with at least two visuals that came intuitively to us. Duplicates emerged as we combined our efforts and began the filtering process. For example, a variation of the 'graph' or network diagram was shown on everyone's sheet. When we reached the categorization phase, we realized that we primarily generated methods of visualizing score and topic associations. To further granularize, the topic association visuals seemed to answer (1) what were the topics about, and (2) which words are more associated with one topic versus another (comparing topics). We took this realization and decided to focus on creating visualizations to answer these two questions, along with visualizing scores.

We started focusing on our specific ideas, and considered the kind of interactions, filters, etc. we could add to each one. As we progressed in our design sheets, we realized some of the variable mappings were not necessarily optimal. For example, using the network to display relation of words to topics became computationally infeasible after further requirements gathering. Size of the 'word' bubble isn't the best way to represent how useful it is to the topic. Moreover, since topics will have some probability for each word, there will be links to each word from each topic.

We can fix this by adding a probability threshold for showing a link, but as we considered the complexity of creating that particular visualization, and the lag it will cause to all other plots, we decided to remove it from the final realization.

The remaining ideations were refined and coalesced into a final display consisting of multiple coordinated views. The final organization will provide users with functional interactions to both filter data and overcome occlusion through cartesian distortion. In the end, we found this realization to have the best balance between simplicity, functionality, and graphical excellence. The FDS method provided a guide to bring our individual thoughts together and develop them into a concrete and attainable visualization. It further helped us identify the most important concepts across the team. We also learned the benefit of generating a prototype using methods completely abstracted from a digital interface. Doing this gave us an extra level of creativity that was essential to producing our final ideation.

Usability Testing and Results

Basic Evaluation Plan

The goal of the visualization is to visualize the results of a topic model on the Reddit jokes dataset. The information we want users to be able to explore are:

1. what the topics of the Reddit jokes are;
2. which words are more associated with one topic versus another;
3. which topics tend to have higher-rated jokes.

We chose to observe tasks that highlighted these goals.

Tasks

We first introduce the user to the page with a brief background. Mainly, we discuss the data, and ask the user to pretend they were assigned a job to find out what the topics are about, and which topics tend to do better in the Reddit community. We then asked the users to perform a set of tasks, and followed with a feedback session.

The requested tasks were to:

1. explain what topics 1 and 3 were about;
2. explain which words were similarly used between them;
3. which of these 2 topics had higher scores on average.

Since our prototype wasn't yet functional, we used a low-fidelity prototype similar to that shown in below. The only difference is we used actual words between the two topics, and conducted our interactions manually on paper.

We took notes as they interacted with the layout, and concluded with a survey to explain their experience and asked for any suggestions to improve the current design.

The usability goal of the visualization is to be functional enough for technically-inclined data scientists, while remaining intuitive enough for the lay user that may be interested in exploring Reddit joke topics. So, we decided to use this evaluation plan on both data scientists and friends.

Users

Non-technical friends

A large portion of our target user base come from non-technical backgrounds. Their focus is not visualizing how well an LDA topic modeling algorithm performs but quickly exploring the data to see the various words, jokes, and topics. These users were selected family members and friends who have little to no knowledge of machine learning or visualization techniques. They made it clear which parts of our visualization were unintuitive or needed further explanation.

The Data Scientist

The final user was a data scientist working closely with the overarching problem of predicting Reddit joke popularity using machine learning models. Naturally, being able to explore these topics effectively will be important to them. This is where the functionality will be put to the test.

Evaluation Results

The overall results were promising, with many great suggestions made by the users. We also witnessed an effective form of indirect feedback from the questions users asked. This gave us a great indication of what did not come intuitively from the design and labels.

Before we discuss the specifics of the feedback, we will summarize some of the discoveries for each task.

1. *Explain what topics 1 and 3 are about?*

All the users jumped straight to the bar plot when asked this question. They read through the words and quickly came to the conclusion that topic 1 was about racial jokes, and topic 3 was about girlfriends and/or kinky jokes. Some navigated to the joke list, only to be disappointed that the jokes aren't necessarily connected to a topic, unless a word is highlighted/filtered.
Most questions asked here were regarding what the length of the bar indicated. We had labeled it 'Probability', but the non-technical users were understandably not satisfied with this answer.

2. *Which words are similarly used in topic 1 and topic 3?*

The goal of this task was to test how easily the user would realize that words lying at the 45-degree line of the scatter plot were those that were used between both topics consistently. This seemed to be a bit more difficult, particularly for the non-technical users. Nonetheless, with little explanation on what the scatter plot represented, all the users came to the determination relatively fast.

3. *Which of these two topics had a higher score on average?*

This task seemed to be the easiest. All users identified the correct topic based on how we arbitrarily decided to draw them, which was made obvious to not leave any doubts on who the

clear winner was. The only struggle shown was the interpretation of the axes of the plot, indicating that they needed to be better labeled.

Feedback

The feedback had many positive notes that were constant among all users. The users all liked the ability of using several coordinated tools to conduct their tasks, including scatter plots for comparing, bar plots for word-topic associations, density plots for score assessments, and a list of jokes to explore. The effective use of colors to distinguish between the topics seemed to be a favorable feature, as well as having details-on-demand when hovering over a word.

We asked about the use of length as a means of encoding word-topic associations, and the consensus was that they preferred it to any other means of encoding. We also asked about the encoding of the word 'scores' (based on average score of jokes using that word) using size on the scatter plot. This also seemed favorable, and they highlighted that even though the exact quantity may not be obvious from the plot, the detail-on-demand feature was enough to provide the needed functionality.

Another positive note was the addition of the 45-degree line on the scatter plot. This especially helped with task (2) – comparing words similar between topics – as it made it easy to distinguish the separation of words between topics. The actual goal of this was to minimize the data-ink ratio. If not for the 45-degree line, we would have to rely on a grid, when in fact all we cared about was identifying separation, and not necessarily the absolute quantities associated with each word.

Not all the feedback was positive. The users highlighted several critical flaws that we had overlooked. Some of the critiques were related to ambiguity, and others related to increased functionality.

One of the ambiguous critiques came of the non-technical users. The meaning of the length of the bars in the bar plots isn't clear unless you have some background in topic modeling. Saying it's a 'probability' is not necessarily helpful either. This is a challenging problem, since the intent is not to provide a user with a lesson on topic modeling, but rather to allow them to explore the data, which by no means requires that level of rigor. We decided that a way to remedy this may be to use 'Affinity' instead of 'Probability' on the bar plot axis. A technical user will easily understand that they are the same, and a non-technical user will not have to be bothered with understanding where the length came from, but rather what it represents.

Not surprisingly, many of the functional critiques came from the data scientist. One mentioned that it would better if they could select a word to 'hold' the filter, instead of simply relying on a hover-based filter. Another critique was on the method used to filter the jokes. Initially, without any words filtered, the single joke list was to display the top jokes, and not necessarily jokes associated with the topics. A suggestion made was to split the joke list in two – one for each topic, and then order the jokes based on most relevant to the topic, then by score. The data scientist

mentioned that being able to read the jokes most associated with the topics would be a helpful way of determining what the topic was about without having to rely solely on a bag-of-words approach.

Another great suggestion made to increase functionality was to have a 'blend' option for the topic-score density plot. The intent is to show if jokes that have a mix of certain topics tend to do better. For example, maybe jokes that have a mix of topic 1 and topic 3 do better than those with topic 3 words alone. This level of functionality would be a great addition to the current design, and is worth adding if time permits.

An additional feature suggested was to have a way of looking at the score distribution among all topics, so that they can all be compared at once. This would be a great way of allowing a user to decide which topic to transition to. Clearly, users would likely transition to either ends of the spectrum.

We decided to add a toggle to the density plot to instead display a histogram of the average score for each topic. That way a user can decide which topics to explore next. This idea of transitioning isn't one we had previously considered in our design. We initially thought of a sequential exploration process, but with 50 topics this is clearly not possible. So we now have to ask the question – why would a user want to look at a particular topic? Moreover, why would they want to specifically study and compare two different topics? Using the distribution of topic average scores is a great feature to add since it gives a user a reason to keep going.

One problem we continued to have during the development portion was regarding the default color choices for topics. We originally thought the best color choices would start with a bright saturation to get the most differentiation between words in the bar plot. This continued to cause problems in the density plot, however, making it difficult to distinguish between topics. The usability test proved to us that this was possible with a much lower saturation than we had expected. (Stone, 2017)

The usability evaluation provided more effective feedback than we could have imagined. Through it, we were able to highlight the good, the bad, and the ugly parts of our design. Realizing where the design was ambiguous in nature helped our visualization become usable to wider audience, and will undoubtedly make it easier for both technical and non-technical users.

Not only did the evaluation expose new features that we haven't considered, but it also revealed a flaw in how the joke lists were being displayed. This is a critical flaw in the design that we had overlooked.

What's most impressive is that the evaluation opened our eyes to another level of user experience that we did not consider at all – the experience of exploring the next topics. And most importantly, it exposed our own biases throughout the design process.

Visualization Implementation Process

We decided to implement the visualization using D3.js, primarily because we wanted to experiment with, and implement techniques like Cartesian Distortion on the scatter plot and dynamic lists to make exploring the jokes easier.

Most of the work relating to keeping the UI state in sync with what's on the screen is handled by the React.js library. (https://reactjs.org)

The app can be split into the following major components:

- *Main app: (App.js*) - The controller for managing the state of the app and keeping up with user interactions. This component creates instances of the following components and provides them with appropriate data.

- *Graphs: (BarChart.js, DensityPlot.js, ScatterPlot.js, TopicBarChar.js*) - Each of these files implements a type of graph using D3. These files are included from App.js which provides them with a subset of the data corresponding to the topics and filters that the user has selected.

- *List Components: (JokeList.js, Lists.js):* These files contain implementation for all the joke lists in the visualization. There are two lists at the bottom which show the jokes contained in each topic, and one list that slides out from the right side of the page when a work is clicked on in the one of the bar graphs.

Most of the graph components are completely self-contained, and can be instantiated multiple times with different data sources. For example, the BarChart.js is a single component that handles both the bar graphs corresponding to topic A and topic B.

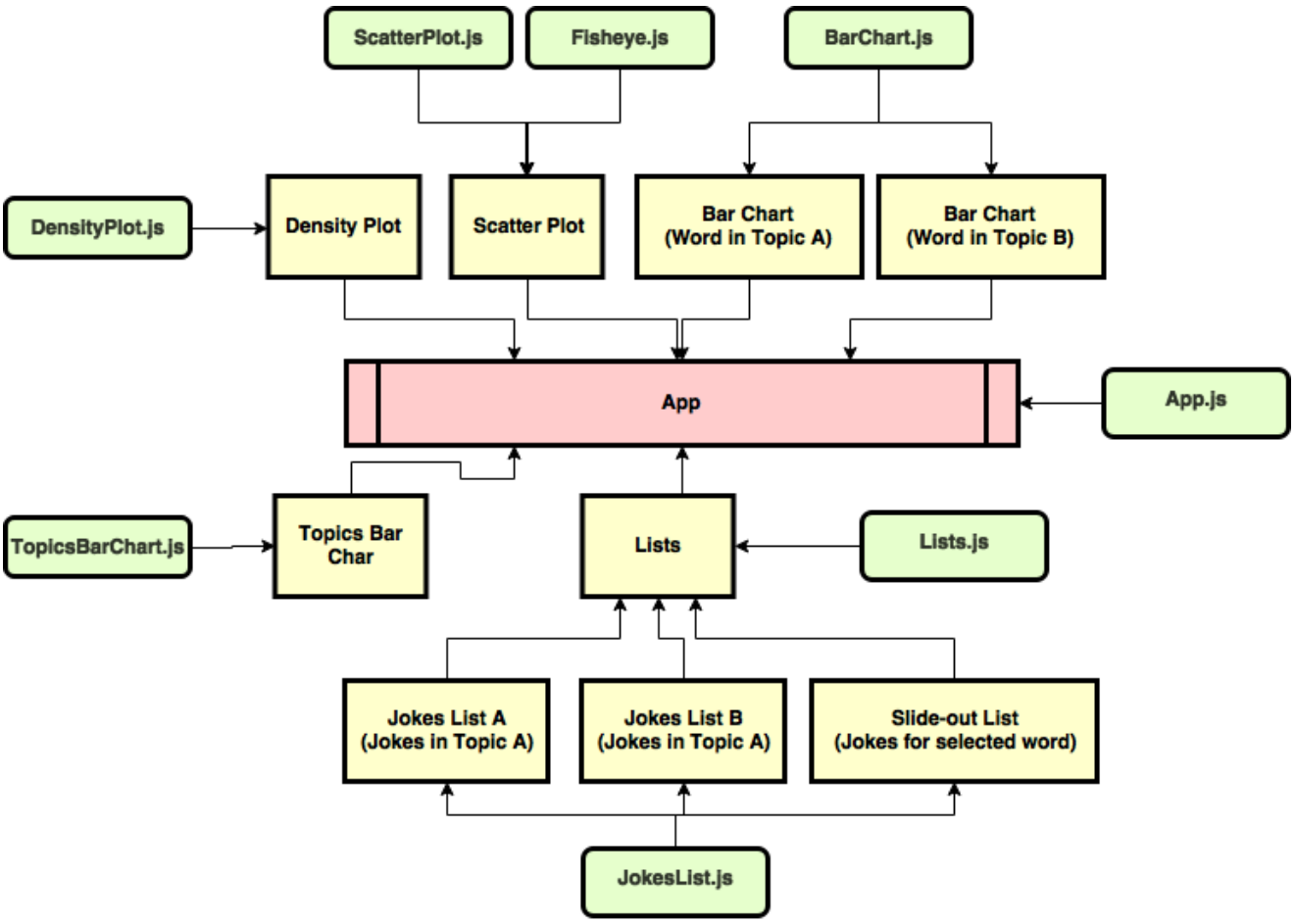

Figure 5: Hierarchy of components and files

Green blocks in the flowchart above represent the source files in the repository. They are reusable components throughout the application architecture.

Yellow bocks represent an instantiated representation of the class defined in the source file.

Appendix A: Final Visualization and Screenshots

Figure Final visualization. The final visualization

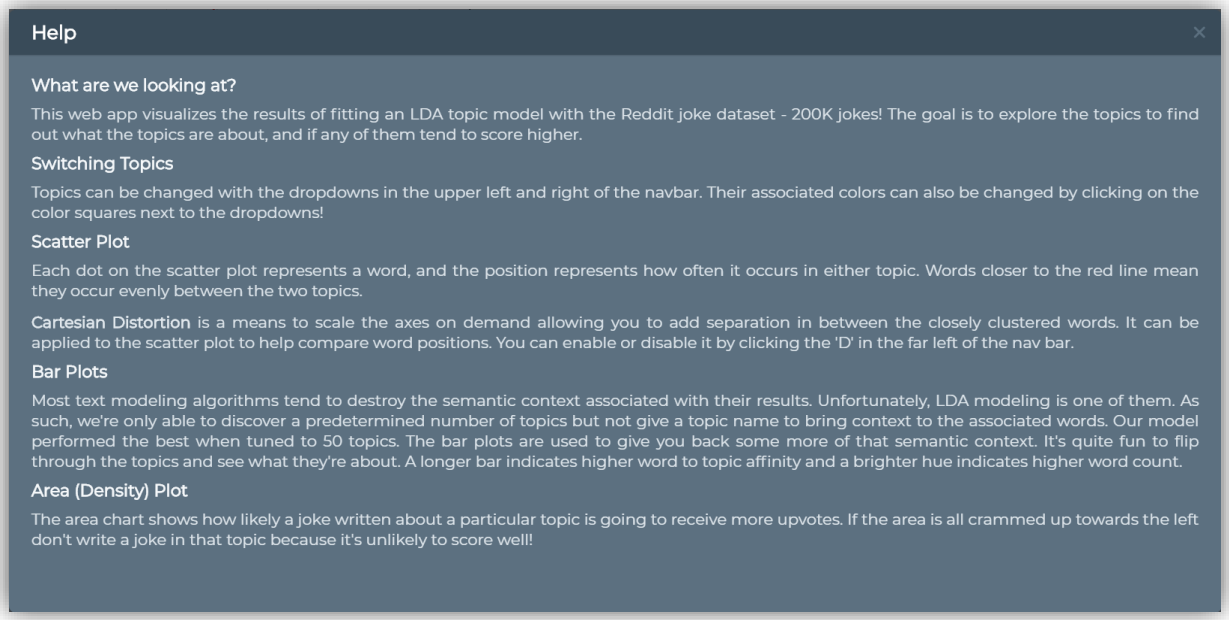

Figure 6 Help dialog. Was a crucial addition in response to our usability study. But further user testing revealed that it was not as intuitive or as accessible as required. Further view space adjustments were made to provide the user with more detailed information. These figures are shown below.

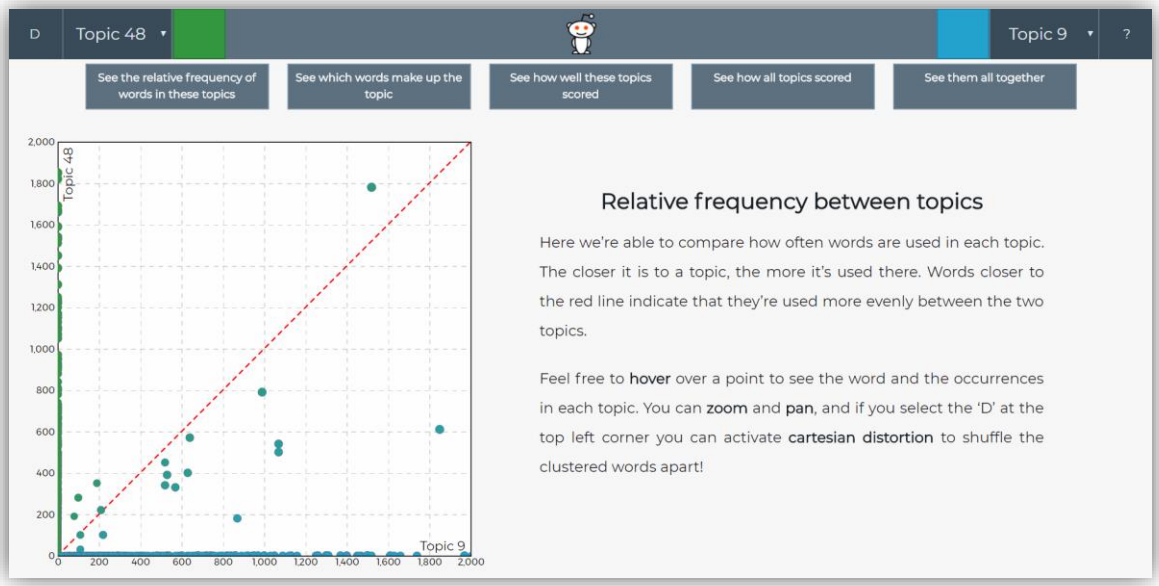

Figure 7 Extra information for the scatter plot

Figure 8 Extra information regarding the bar charts

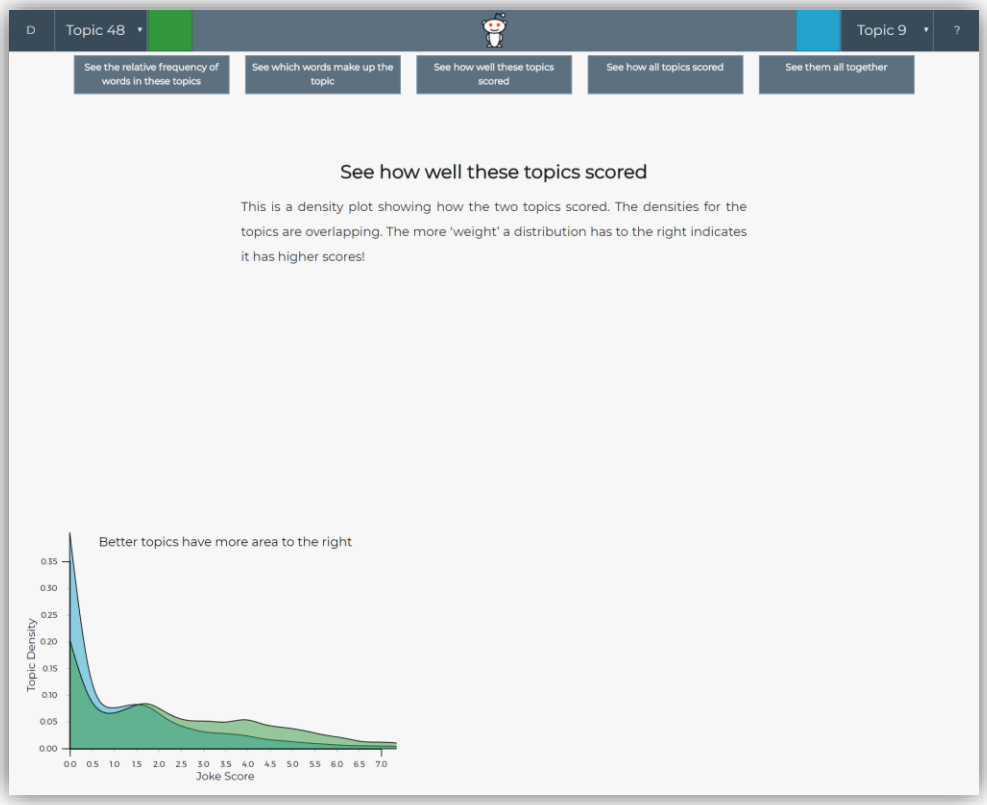

Figure 9 Extra information regarding the density plot

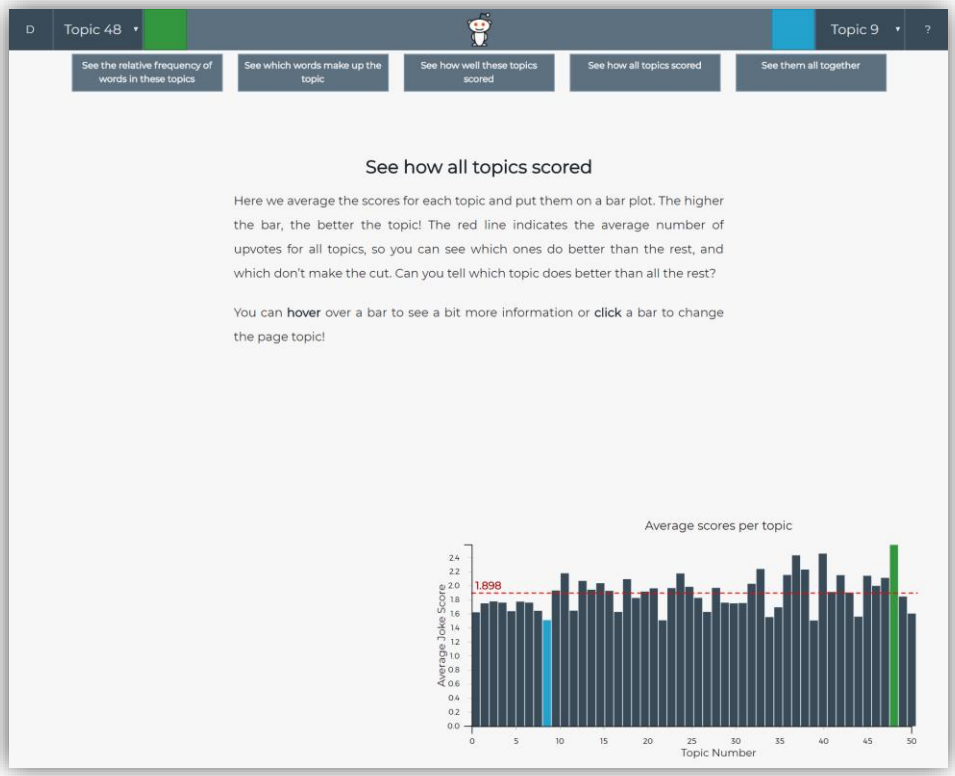

Figure 10 Extra information for the topic bar chart

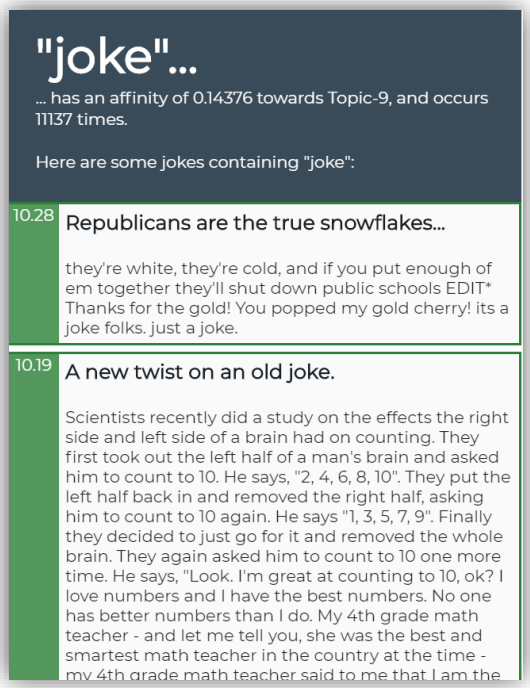

Figure 11 On demand sidebar joke listing

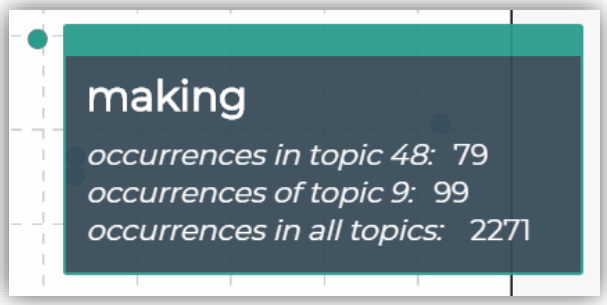

Figure 12 Scatter plot tooltip

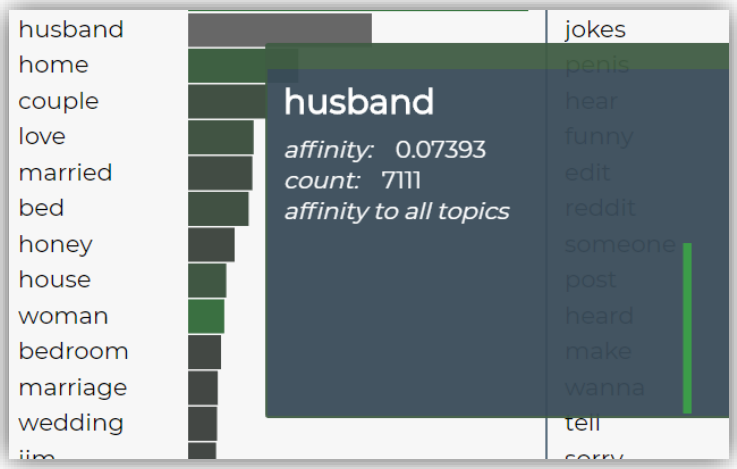

Figure 13 Bar chart tooltip hovered a word that only appears in its most associated topic

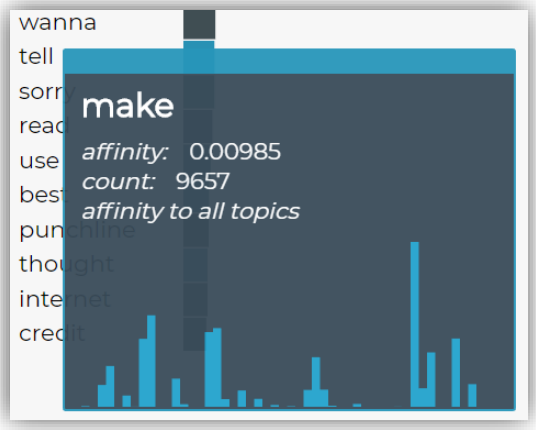

Figure 14 Bar chart tooltip hovered a word that appears in many topics

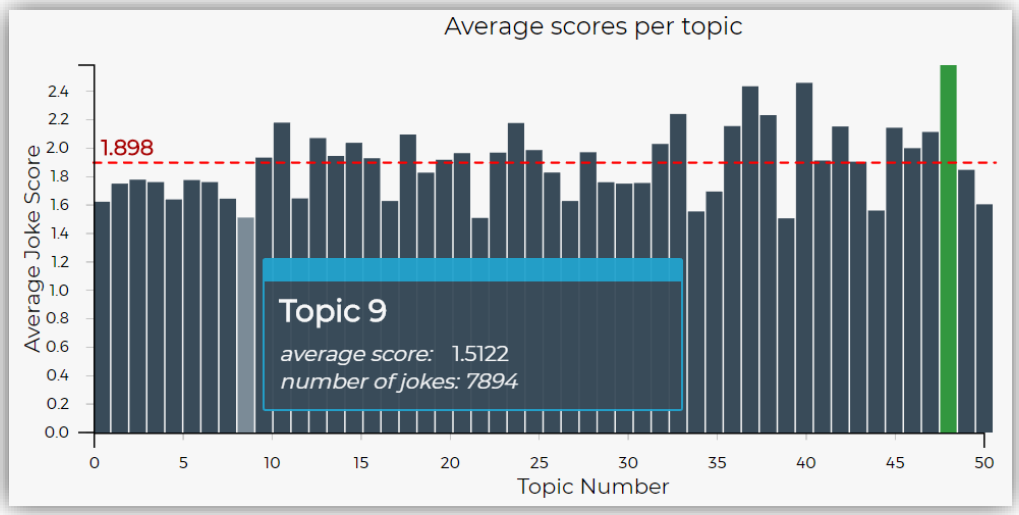

Figure 15 Tooltip hovered over the topic bar chart

Appendix B: Design sheets

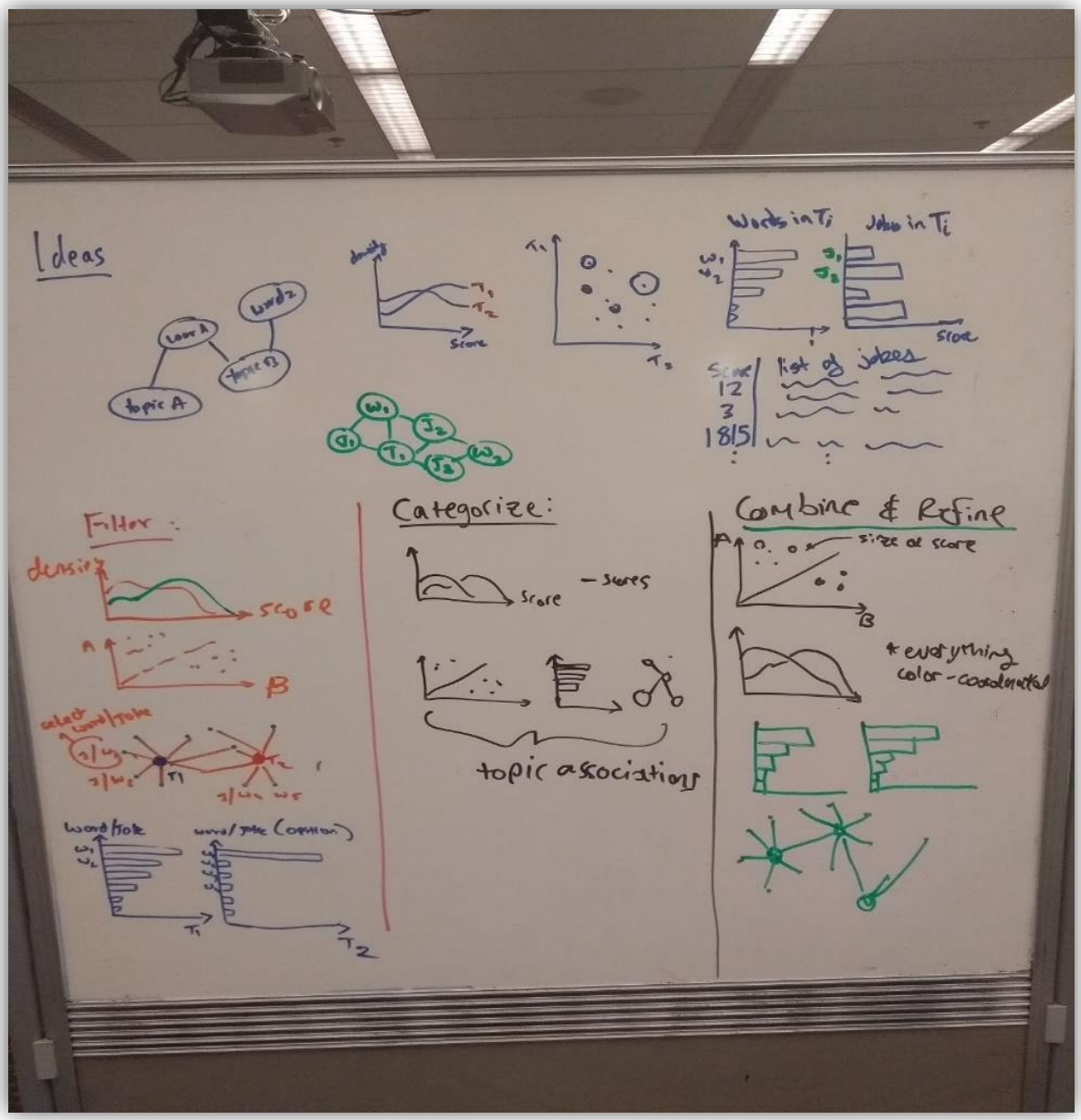

Figure 16. FDS#1 - The initial team brainstorming compilation

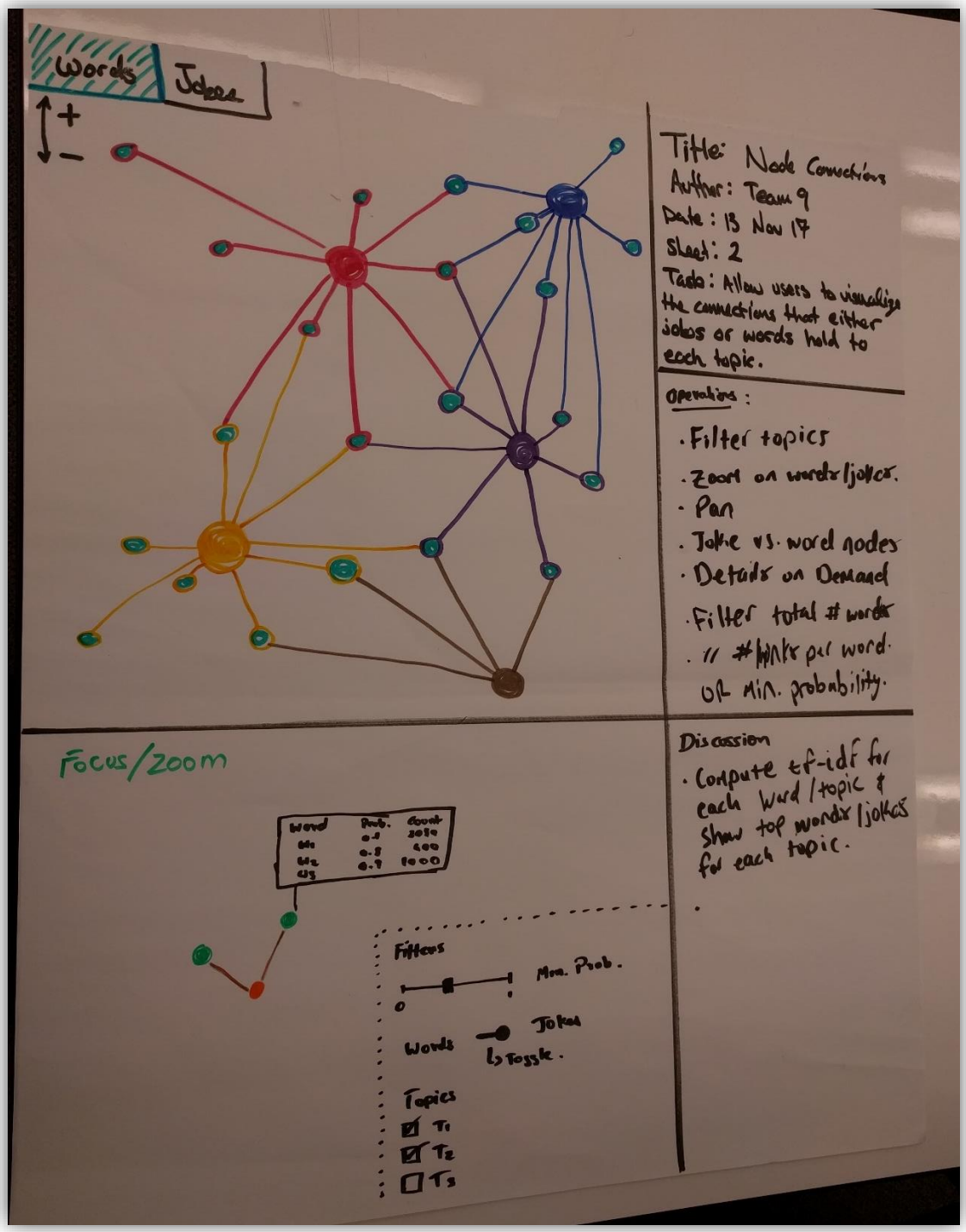

Figure 17 FDS#2 - Node connections

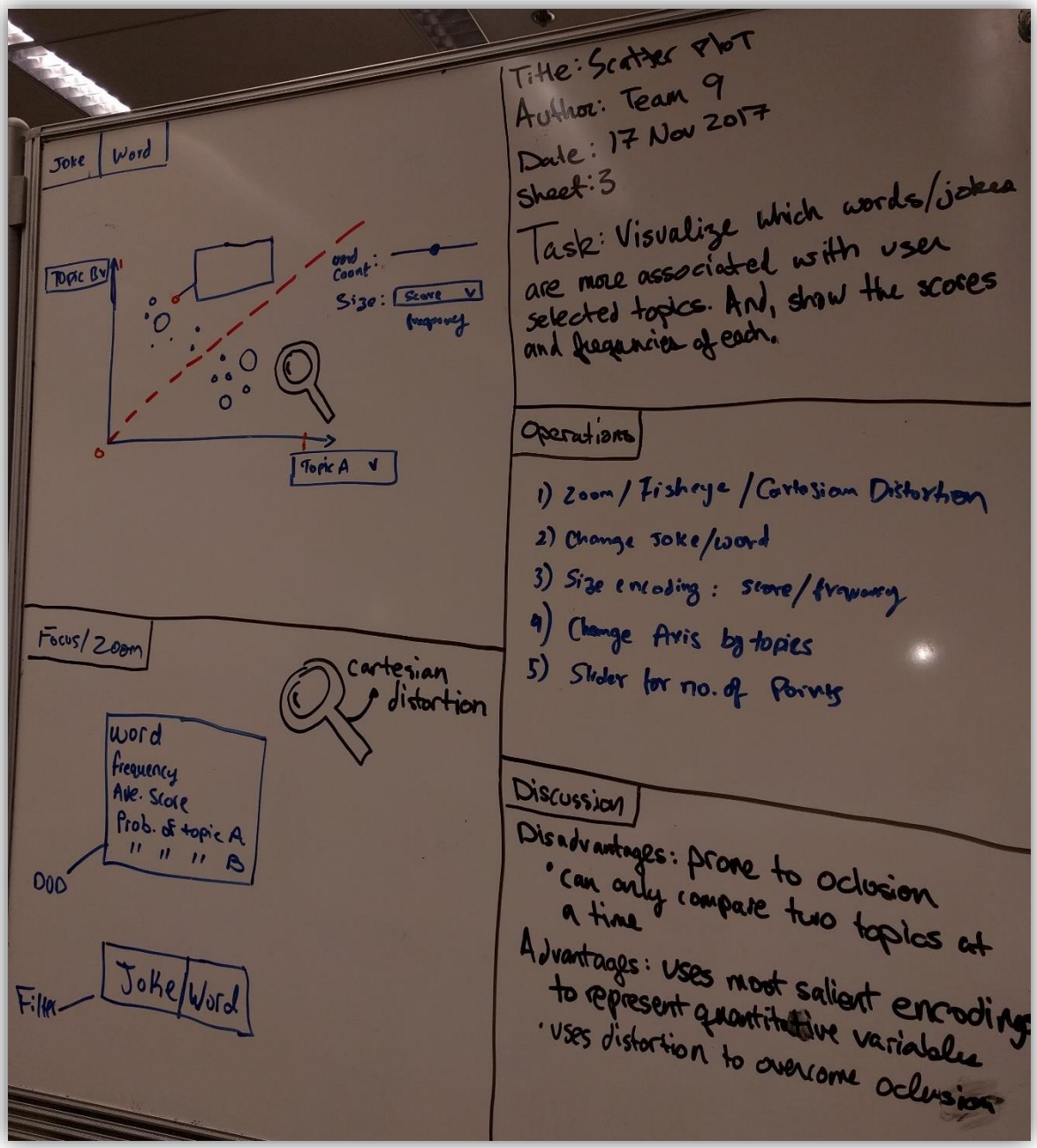

Figure 18 FDS#3 Scatter plot

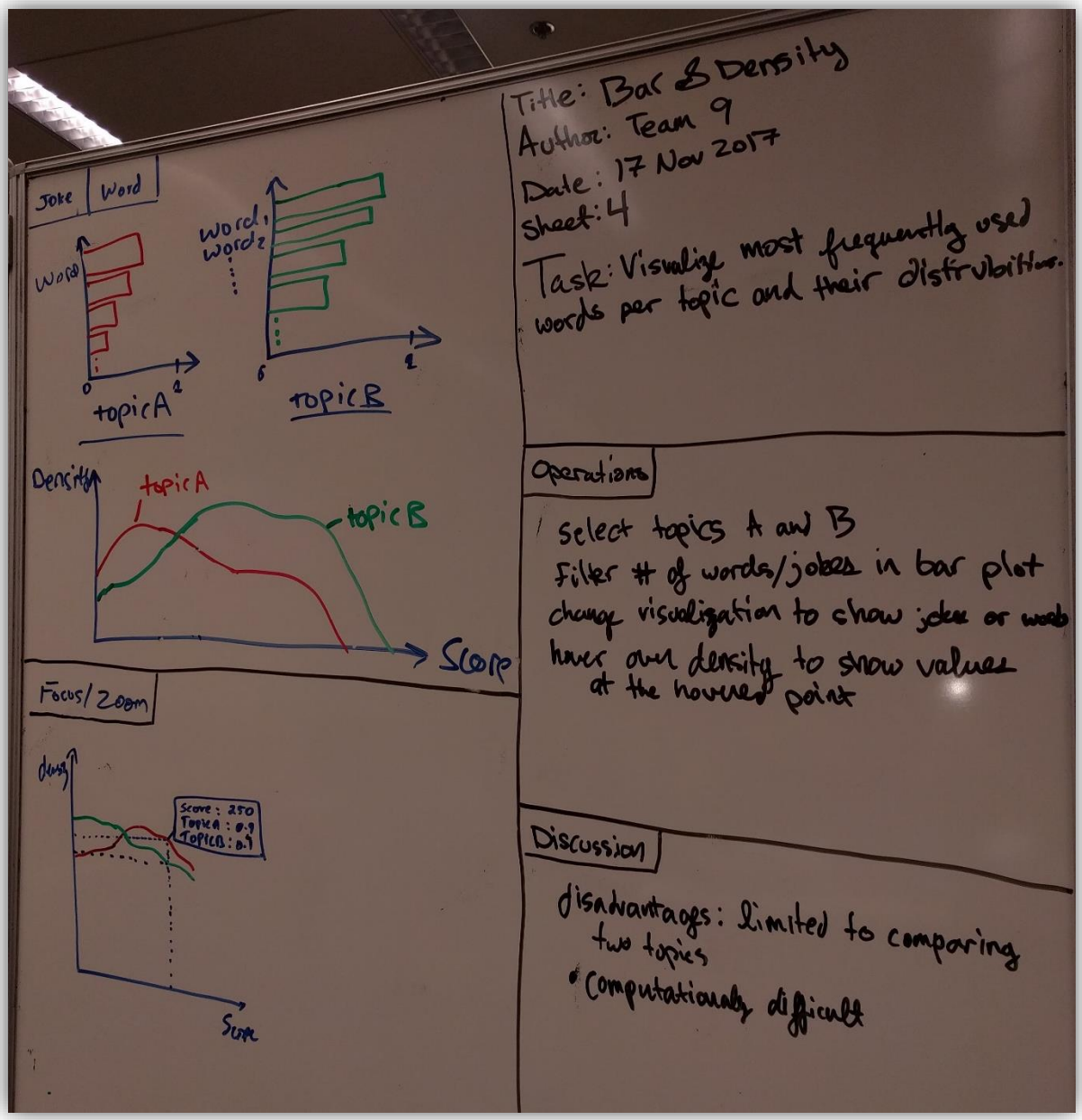

Figure 19 FDS#4 - Bar plots

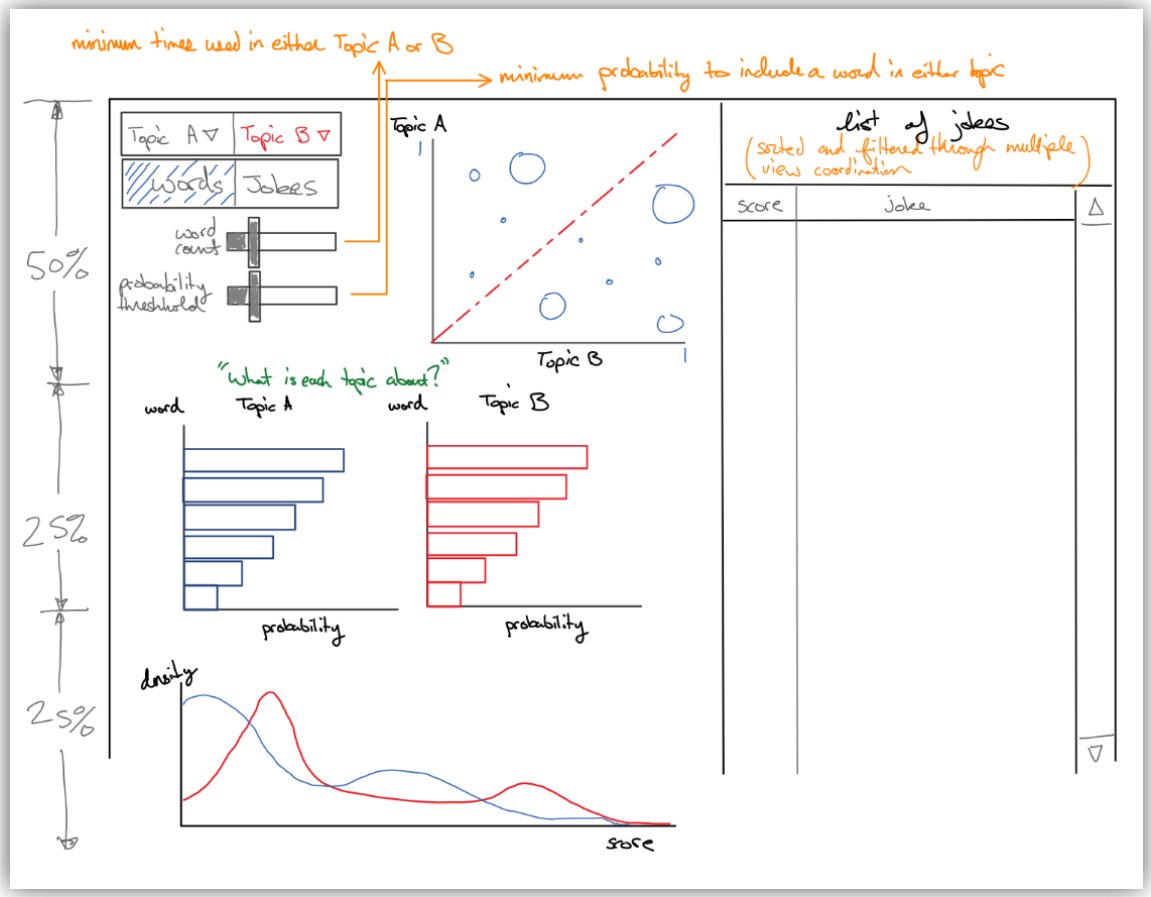

Figure 20. FDS#5 - The final design ideation

Appendix C: Low Fidelity Prototype

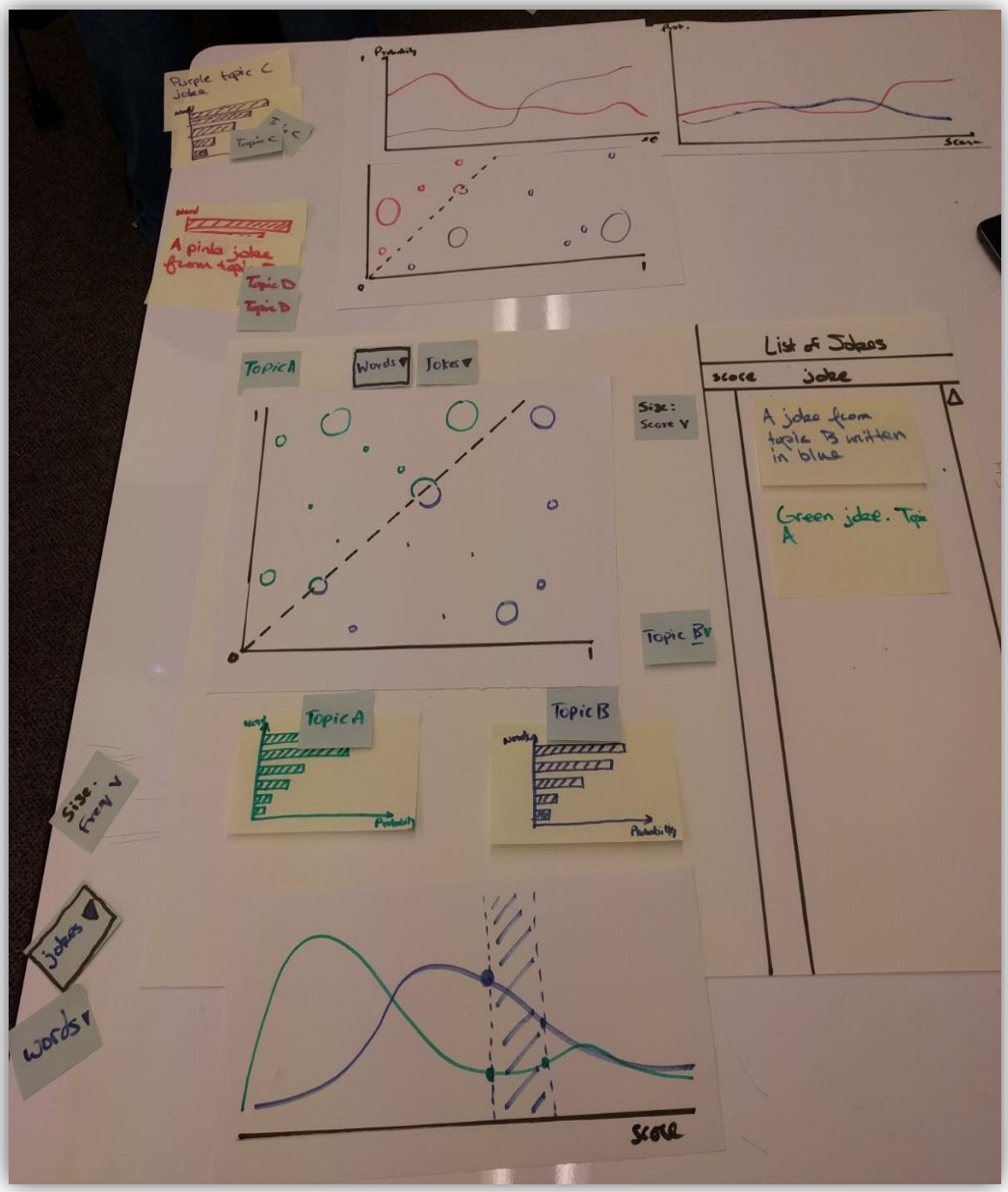

Figure 21 Low fidelity prototype

Appendix C: Links and Bibliography

Final visualization: http://lkwaninger.ddns.net/

GitHub repository: https://github.com/lukeWaninger/HCDE556

Bibliography

Casesandberg. (n.d.). Retrieved from casesandberg/react-color: https://github.com/casesandberg/react-color

D3. (n.d.). *D3.js*. Retrieved from D3.js: https://github.com/d3/d3/wiki/Gallery

Facebook, Inc. (n.d.). *Create React App* . Retrieved from facebookincubator/create-react-app: https://github.com/facebookincubator/create-react-app

Facebook, Inc. (n.d.). *React.js Library*. Retrieved from https://reactjs.org

Few, S. (2009). *Now you see it.* Oakland, CA: Jonathan G. Koomey.

Few, S. (2012). *Show Me the Numbers.* Burlingham, CA: Jonathan G. Koomey.

Jonathan C. Roberts, C. H. (n.d.). Sketching Designs Using the Five. *Visualization and Computer Graphics*. IEEE Transactions on.

Material UI. (n.d.). *Material UI*. Retrieved from mui-org: https://github.com/mui-org/material-ui

Palantir, Inc. (n.d.). *palantir/blueprint*. Retrieved from BlueprintJS: https://github.com/palantir/blueprint

Stone, M. (2017). Expert Color Choice: How and Why. Tableau Software Inc.

Tufte, E. (2010). *Beautiful Evidence.* Cheshire, CT: Graphics Press LLC.