IDEALvr

Final Project Report

Team: Luciano Biondi, Omavi Walker, Dagmawi Yeshiwas
Client: Mohamed Magdy
Course: CS 4624 Multimedia, Hypertext and Information Access
Virginia Polytechnic Institute and State University
May 1, 2016

Table of Contents

Table of Tables

Table of Figures

Executive Summary

This report provides an explanation into the implementation of the IDEALvr application and an analysis of the methodology to create the system[1]. The methods of implementation include 3D virtualization onto an Android application using the game engine Unity and Google Cardboard 3D virtualizer. The methods of analysis include gathering metadata from the IDEAL database and creating new methods of analysis of pre-collected datasets. Other analysis includes understanding the different limitations and advantages of using Google Cardboard and Android to develop the user experience.

Our clients Mohammed Magdy and Dr. Edward A. Fox served as helpful resources in creating the requirements for this project and providing feedback on a weekly basis to change or continue on development paths. The requirements were outlined as (1) visualize the IDEAL collections in a meaningful way, (2) create a working interface for exploring the collections and (3) create an application that allows users to draw conclusions from IDEAL data.

The goal of this project was to provide an easy to use interface that allows researchers to visualize collections in a way that increases understanding and allows them to draw conclusions from the data without having to read through massive collections of tweets. We created a hierarchical structure that allows users to filter collections by categories. We developed a frequent word analyzer to parse through the collections and generate a list of the most frequent words in a collection which is also visualized in the project. The virtual reality user interface allows for full immersion and interaction with as many IDEAL collections as the user desires.

The IDEALvr project was created with Unity and developed to work on Android devices that are supported by Google Cardboard. Work on a web application was attempted and issues are documented in the included files. Future work and plans are also included in the attached reports.

Overall, the IDEALvr project was very successful. It provides a way to visualize IDEAL collections in a way that allows for meaningful analysis and conclusions to be made. The code can be found at https://git.cs.vt.edu/dagmawi/ideal-vr.git.

1. User Manual

1.1. Introduction

Since the recent rise of virtual technology, we have seen it gain massive popularity in the gaming industry with corporations developing their own virtual reality headsets. However, there are still numerous other applications for the technology and one of them is data analytics. Virtual reality can open up new dimensions and show us new ways to look at the same data to try to find more information from analyzing it in different ways. To do this, we took one of the largest collections of data we could find to create a 3 dimensional view of it to transpose information in new patterns to see what can be analyzed from the collection.

IDEAL is the Integrated Digital Event Archiving and Library developed by Edward A. Fox, Kristine Hanna, Andrea L. Kavanaugh, Steven D. Sheetz, and Donald J. Shoemaker at Virginia Tech[2]. The goal of the project was to aggregate specific hashtags of updates on social media and the general web, including news and governmental information to consolidate an immense amount of data for analysis. They then created an information system to include related metadata and knowledge bases to support access to the content that is available and related to the information from the specific hashtag updates collected. To make this happen, the IDEAL team integrated the Twitter API and were able to pull tweets according to different keywords scanned. This created their database to create the system of information for the knowledge base.

Taking the concepts from IDEAL we wanted to create a similar system, but with a new dimension of understanding the data that was collected. We are integrating their data into a visualization engine using Unity. The IDEAL team developed their datasets to be able to be exported into JSON strings. This platform provides a way to pull their data using JSON strings. Using this technology we want users to be able to experience a visual aspect to the information rather than just reading it. We wanted to create a way, using virtual reality, to bring the information to life.

Google has recently developed a way that many people can commonly use virtual reality at a low cost. They have created a virtual reality headset that is accessible to anyone and provided a way for their Android technology to easily integrate to applications that want to be developed for virtual reality.

Using these tools we will create an application that will immerse the user into the data and map out the connections of the datasets to help users identify new patterns and analyze in new ways. Using data visualization techniques and Unity based virtual reality, IDEAL VR will construct a digital mapping of tweets. Using this mapping, users will be able to interact and divulge into queries by focusing on specific updates. The proposed system will be an android application using Unity to provide data visualization of tweets.

1.2. End Users

The users for this application are the IDEAL team of researchers and developers who will use this application to visualize their tweet collections but anyone in the general public willing to work with the IDEAL team will be able to use this application to conduct research on the IDEAL teams tweet collections. When considering the end user, creating a virtual reality as a user experience required thinking of fundamentally new ways to build for the user. Unlike other user experiences, we will have to work constantly on the stability of the visualizations to provide optimum comfort. There are two types of end users that will benefit from this application: those who want to view information and those using the application for research purposes. Ultimately, the end user is an information seeker and the most important aspect will be designing for a user so that the data is easily acquired.

1.3. Use Scenarios

There are multiple use scenarios for IDEALvr. The main use scenario will be that of using the data visualization to construct new analytical data from the metadata presented to the user. This is the main goal of the virtualization and the main idea driving the use of this new way to view information and data collection. There is also the scenario where data will be indexed for usage and knowledge rather than for analytical purposes. This will primarily be the case with end users who need to retrieve data from a specific collection, but do not intend on further analysis.

1.4. Content, Information, and Data

The purpose of the IDEALvr project is to dynamically display and visualize Twitter data for users in a virtual reality environment. By displaying the information in such a way as to make it easier to draw conclusions and see connections between the data, our VR visualization makes it easier to analyze and interpret patterns. The data to be used initially will be static collections of tweet metadata collected by the client. The tweets will be divided into different collections depending on what keyword or phrase was used to retrieve them.

These keywords (or key phrases or hashtags) will be unique identifiers to differentiate between different collections of tweets and metadata. Each collection will be built using a specific keyword and will be accessed with that same keyword. Each of these collections will belong to a hierarchical category which will be used to further categories the collections.

1.5. Interface

Visualization of data is the biggest part of the IDEALvr project. The goal is to create an interface that facilitates the viewing, searching and filtering of large collections of data. The benefits of immersing a user into an environment and displaying data through different techniques has been studied and proven in studies conducted by Dr. Bireswar Laha of Stanford University[3].

The interface will be provided to the user as a visualization on a headset display which allows for a full spherical range of vision. The user will interact with the data by clicking on the components that will reform the visualization and allow for further interaction. The user will initially be show an organized display of categories. By clicking on a category, the display will be reloaded with an organized display of keywords representing collections. Clicking on a keyword will reload the display to an actual word cloud with the most frequent words that appear in the tweets of the specific collection selected.

The data in the virtual environment will be displayed in several ways. The project will initially display the categories of the collections in a word cloud spanning 360 degrees and surrounding the user in a spherical coordinate system. The data will be displayed in such a way as to avoid keywords from overlapping each other or from being too out of the way that it would take physical strain for the user to view a word. **Figure 1** displays how the word cloud will appear to the user. In this interface, the user will remain stationary but have full control of their field of view to be able to examine different parts of the word cloud at will.

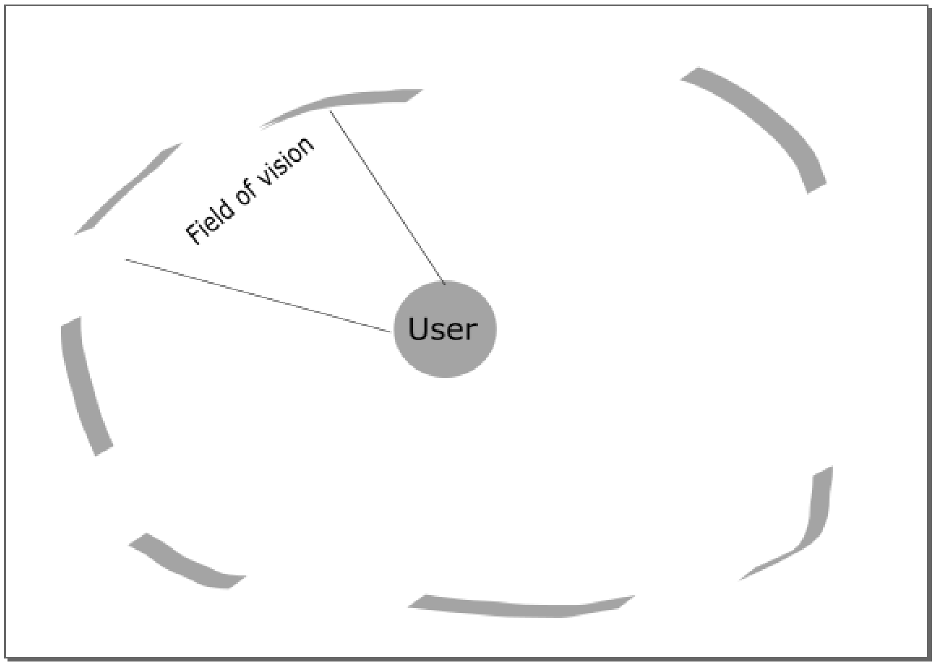

Figure 1. Top view of placement of words in cloud with respect to user view range

The main visualization technique that will be employed at this stage is to generate the words in the cloud to have different sizes and be organized in a user friendly way. The categories will be displayed in different sizes, depending on the percentage that their respective collections represents in the total amount of tweets aggregated in all collections in the category from the database built by the application. The higher the percentage, the larger the keyword will be displayed.

The category and keyword collection displays will be organized in a way such that the user of the application knows where to look for a certain location. These views will display text in an alphabetized order from left to right and will show larger collections in the middle and with smaller ones as the displays go up and down.

The initial view will show the categories. When the user focuses on a certain category (which will be indicated by highlighted text) and clicks the application, the word cloud will be automatically reloaded to show the collections that belong to the category which was clicked. **Figure 2** shows an example of how the word cloud will look from the user's point of view for the collection view. The category view will have the exact same visualization as **Figure 2**. The user will be able to tell the relative sizes of the collections without having to read and compare numbers. Careful consideration will go into determining a minimum and maximum size so that all keywords are still readable.

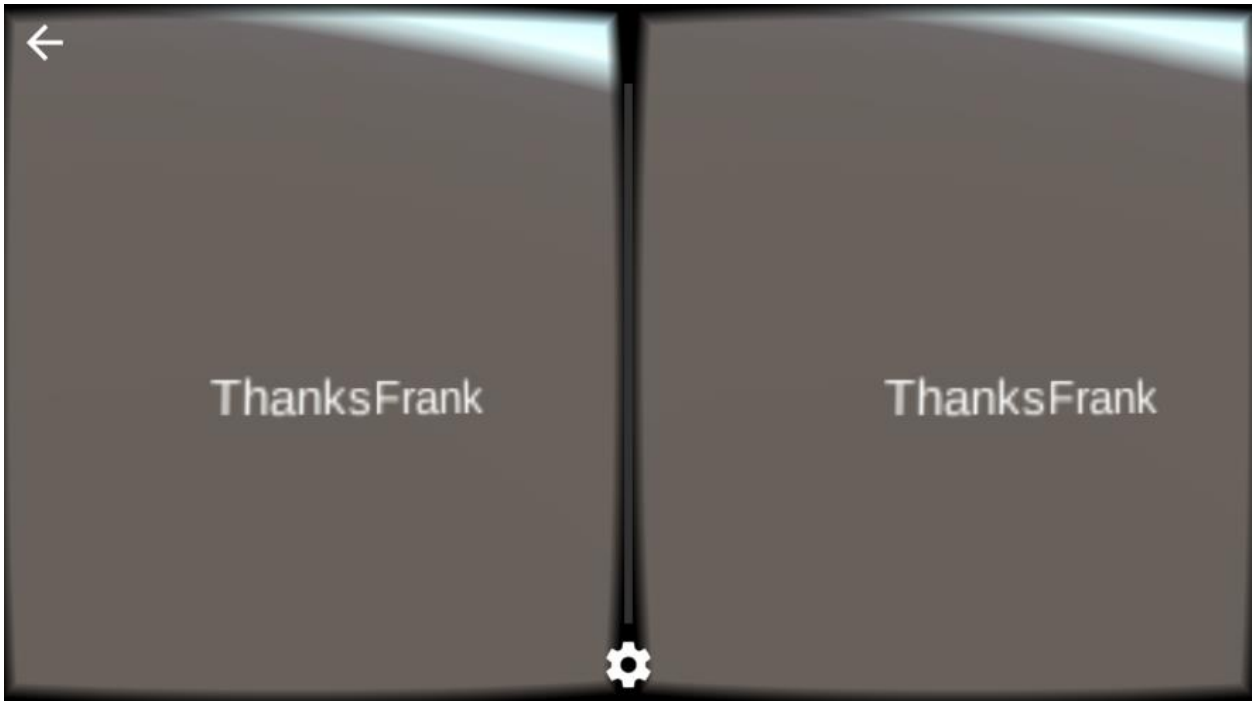

Figure 2. Collection keyword display from user viewpoint with varied word size

The last interface available to the user appears when the user selects a collection from the collection cloud display. In this stage, the user is presented with a word cloud of the most frequently appearing words in the tweet collection. These words will be placed randomly in the word cloud display but will also have varying sizes with larger words representing words that appear more frequently. This will allow the user to determine what the tweets in this collection were about. **Figure 3** shows how the frequent word display will appear to the user.

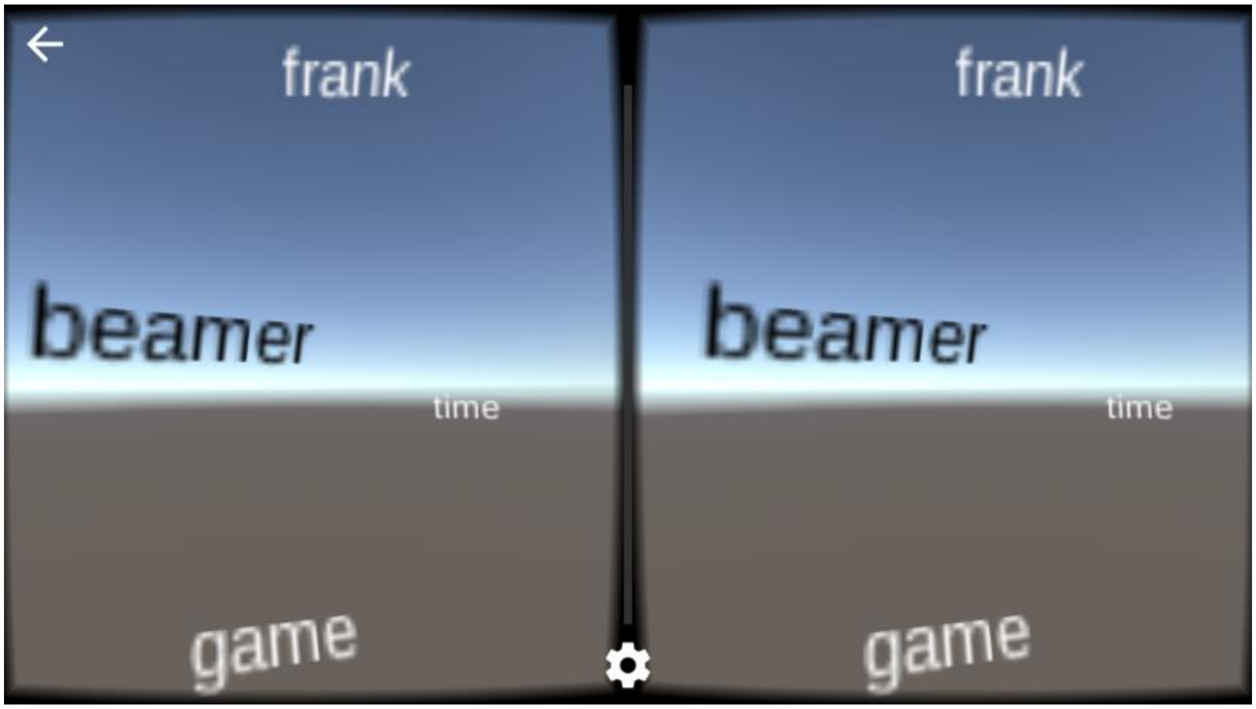

Figure 3. Frequent word display from user viewpoint

In every stage of the application, a “BACK” button will be present in a fixed location that the user can click on in any scenario. These buttons will appear in the same location at every screen so that the user knows where they are at all times. The “BACK” button will be available and clickable when appropriate. It will send the user to the previous point of the interface. For example, from the collection word cloud interface, the back button would send the user back to the category interface.

1.6. Software and Hardware Specifications

The project requires the user to download Unity version 5.3.2 on a computer capable of running the software through the Unity website. The user then needs to download the most recent Android SDK or at least 1.8.0 version 60 through the Android website. The hardware necessary for running the IDEALvr application includes a Google Cardboard with any type of button that allows for clicking on the screen (a physical button or a magnetic sensor) and an Android phone with API level of at least 19 running at least Kitkat 4.4 or above. A Google Cardboard can be purchased at http://www.amazon.com/Pro-Compatible-Instructions-Construction-Experience/dp/B00Q1FITMO/ref=sr_1_1?ie=UTF8&qid=1461611771&sr=8-1-spons&keywords=google+cardboard&psc=1.

1.7. Application File Overview

This section provides an overview of the folders and files located in the accompanying ideal-vr.zip folder which contains all of the code and sample data necessary to run the application.

Table 1. ideal-vr.zip File Overview

File/Folder	Description
ideal-vr	This folder is the Unity project. When opening the ideal-vr application in Unity, this is the folder that should be selected.
ideal-vr/Assets	This folder contains all of the scripts and data necessary for the application to function.
ideal-vr/Assets/FormWordCloud.cs	This C# script provides the necessary functionality to read the generated JSON files from the StreamingAssets directory. It also contains the algorithms to calculate the location of the words in the word cloud shape in a virtual environment. This scrip is attached to the Word Cloud object in the Unity editor.
ideal-vr/Assets/TextSelector.cs	This C# script provides the necessary functionality for calculating user sight vectors and word selection and clicking. It is attached to the word selector object in the Unity editor.
ideal-vr/Assets/Cardboard	This folder contains all of the necessary Google Cardboard objects and assets that work in Unity.
ideal-vr/Assets/StreamingAssets	This folder is the hierarchical structured folder that is created by the Parser. It is where all of the JSON data files that the application reads from are stored and found by the application using the FormWordCloud script. It is essential that this folder has a proper structure as defined in section 1.8 and is placed in the Assets folder.
ideal-vr/ParserUnity/src/Parser.java	This Java code parses through the collections and creates the JSON files that the FormWordCloud script reads. It generates the files by reading from a StreamingAssets folder located in the ParserUnity directory. The StreamingAssets folder in the ideal-vr/ParserUnity folder should be placed into the ideal-vr/Assets folder so that the application has access to it.

1.8. How to Run Application

In order to set up the application, the user must:

- Download the project folder and place it in the Unity workspace folder.
- Open the ideal-vr project folder in the Unity editor
- Make sure the Unity editor is linked to the Android SDK by:
    - Pressing **Edit** then **Preferences** then **External Tools**
    - Locating the Android SDK in the text box that says **Android SDK**.

In order to customize the IDEAL collections to be included in the visualization:

- Enter the ParserUnity folder in the ideal-vr folder.
- Create a folder named StreamingAssets,
- Inside StreamingAssets make folders named as the categories from the different collections you want to include.
- Manually drop the collection into its respective category folder that was just created. Run the Parser code from the ParserUnity/src directory in order to generate the proper JSON files that the application will read.
- Copy this StreamingAssets folder into the Assets folder inside the ideal-vr project folder (replace the existing one if necessary).

In order to run the application on the phone:

- Make sure the user's Android phone is connected as a media device to the computer through a USB cable.
- Select **File** and then **Build Settings**.
- Select **Android** in the **Platform** menu and then click **Build And Run**. The Unity application will ask you to name and save an APK file. Keep the default name and select **Save**.
- The application will then automatically detect the user's phone and upload the application and automatically run it.
- Place the Android phone into the proper position in the Google Cardboard to finish the setup.
- The application will be up and functional immediately.

2. Developer Manual

2.1. Design

This section goes over the design and general idea of the IDEALvr application. Refer to section 2.2 for a detailed description of actual implementation.

2.1.1. IDEAL Database

We will be using data from the IDEAL database to build our virtual space. This database contains massive collections of tweets separated into categories and events. The database currently holds over one billion entries spread out from categories pertaining to social, political and global concepts. The data in this database was extracted by querying the Twitter database, using Twitter's APIs. Currently the IDEAL team has developed the code to retrieve these databases.

2.1.2. User Experience

Once the user decides upon the category that they would like to explore, they will select that option and be immersed in a new view containing all of the keywords and hashtags for that category which represent the collections in the category. Keywords differ because while hashtags are purposefully tagged with a '#' by the user, keywords are words or phrases extracted from their tweet. Once a keyword is selected, a new view will be generated. After selecting an option on this screen the user will then view a word cloud of the most frequent words in the tweets of the selected keyword.

2.1.3. JSON Parsing and Printing

JSON (JavaScript Object Notation) is the lightweight data-interchange format that we will utilize for the collecting and passing of information through the application. JSON plays quite a large role in this project. The IDEAL data used for the project was saved as JSON strings in a file for each collection. For a specific collection, such as "ThanksFrank," every tweet collected for that hashtag would be stored on an individual line of the file. Each of these lines contain a JSON string with all of the information extracted for a specific tweet. Determining the number of tweets for any collection is as simple as counting the line numbers for that file. Additionally, the data for the word cloud will be saved as and read from JSON strings. The IDEAL data contains a plethora of information; however, only the keywords or hashtags and the number of occurrences for that specific query are needed for displaying the word cloud along with the correct relative size of each word in the cloud. Once occurrences are determined by line number, the correct display text for the cloud is extracted by using the name of the original folder (for categories) or filename (for specific file collections). There may be many different types of JSON strings stored for different views such as the categorical, event, collection, and date views. Nevertheless, each individual string would only contain the word or phrase that was supposed to be viewed, along with the number of occurrences.

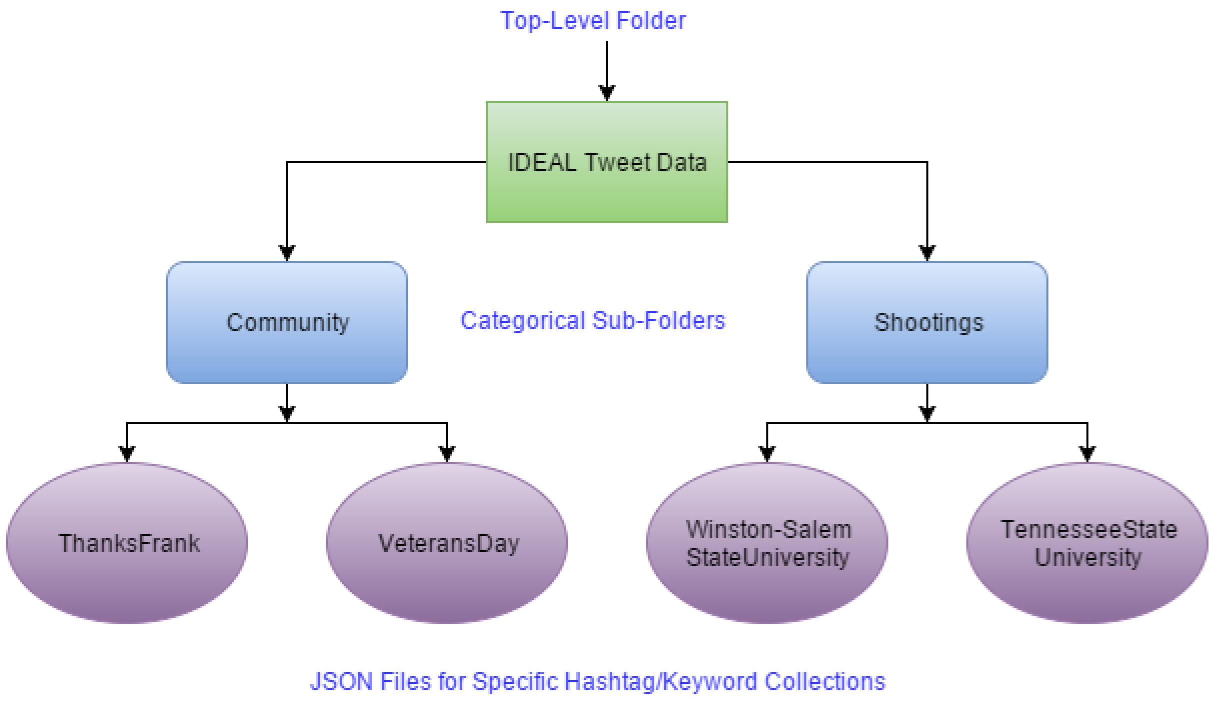
Top-Level Folder
IDEAL Tweet Data
Community
Categorical Sub-Folders
Shootings
ThanksFrank
VeteransDay
Winston-Salem StateUniversity
TennesseeState University
JSON Files for Specific Hashtag/Keyword Collections

Figure 4. File and folder hierarchy for storing categories and tweet collections

Figure 4 shows the entire, general, file hierarchy for the project. Every file containing the JSON strings will be stored in the same folder as whatever information that it is displaying. For example, the original screen with the categorical view has all the categories stored as subfolders in our root folder. The only other file in that root folder contains a file containing all of the JSON strings with the names of those category subfolders and the number of tweets contained in those subfolders. If one were to explore one of those category subfolders, they would find the original JSON files from the IDEAL project. The only other file this subfolder would contain is a file containing all of the JSON strings we created with the names of the keywords/hashtags contained in this category along with the number of tweets in each file.

To achieve this level of organization, our group created a Java program to go through all of these folders and produce the files containing the JSON strings. After creating these files, the program makes sure to save the files in the correct place so that the word cloud application knows where to look for display information. Currently it is still required that someone creates all of the category folders then saves the correct keyword/hashtag collections to the corresponding folder. Everything else from this point is automated. In a future iteration of the project, there will be a way to create these categorical folder automatically, as well as save the corresponding collections to them.

2.1.4. Frequent Word Analysis

There will be a feature to read and process the list of tweets corresponding to a specific collection and build a list of frequently appearing words. The application will process a certain collection's tweet file and create another JSON file which will be of the same structure as the previously generate JSON files. The processing will involve a linear word analysis to determine the most frequently appearing 20 or so words in the collection of tweets, which is a costly process but will only be run once when generating the file hierarchy. Using this JSON file, we will create a separate word cloud next to the tweets view that will consist of the generate list of words. The user will be able to tell what a collection is about and what occurred by seeing the most relevant words at a glance. This feature is where the VR experience could significantly improve, compared to the desktop/smartphone experience that users currently use to find and group the same data.

2.1.5. Unity

Unity will be used for all of the rendering and design of visual information. This engine was a great choice because it allows for easy general development that can then be deployed as an Android, iPhone, or web application. A combination of C# scripts and 3D graphical objects are used to represent the word cloud and its layout. When the application is run, the Unity engine processes the JSON files to output the correct view.

The ability for this application to be produced for so many different platforms is very powerful because most users of technology would be able to access this app with at least one of the options previously stated. Smartphones other than the iPhone and Android devices could easily use the web app to use the same features as the native applications. Even if someone did not have a smartphone, they could use a computer to view the web app or even a tool like the Oculus Rift to still have the full VR experience. The use of Unity has essentially future proofed this application.

2.1.6. Cardboard

The Google cardboard Unity SDK was used to seamlessly convert the 3D world developed in Unity into a split screen version that could be used with Google cardboard. Implementing this capability was as easy as importing the SDK and replacing the standard viewing camera with a Google cardboard viewing camera. The Google Cardboard can then be used to visualize the application as a 3D environment. Cardboard has a button on the side that translates to a user clicking their phone. Cardboard was chosen as the main focus for the project because of the wide prevalence of smartphone owners in the world, the ability to view applications in 3D and the button that allows for clicking. With a $10-35 purchase, a user could easily turn their normal smartphone into a full VR experience.

2.2. Implementation

This section goes over in great detail the steps taken to build the application. Following these steps should allow a developer to implement the project from scratch into its current state.

2.2.1. Setting Up Cardboard Unity Project

The IDEALvr application is developed in Unity specifically for a Cardboard application. The first step of the project is to visit the Unity site at http://unity3d.com/get-unity/download and download Unity along with the Unity SDK. Unity is free for individual and educational purposes. Unity provides an interface and framework for rendering and creating interactive 3D worlds.

This application is going to be developed for the Google Cardboard which already has a Cardboard development kit. In order to download the Cardboard Unity SDK, visit the page https://developers.google.com/cardboard/unity/get-started to download the SDK and a demo application. In a newly created Unity project that was created before, go to **Assets** in the menu bar, then click **Import Package**, and **Custom Package**. Search for the Google Cardboard SDK and select it to import it into the project. This SDK provides the assets and objects necessary to start developing a Google Cardboard application in Unity.

When one creates a new Unity project, an empty world is generated by Unity. Create a **Scene** which will allow the user to place objects into the world. Through this interface, you can create 2D and 3D objects or download and use already existing assets which are objects with properties already defined by Unity or Unity users. Objects can be easily placed into the Unity world by dragging and dropping them into the assets hierarchy window of the Unity project. Every object has a set of properties which can be set or dynamically changed through the use of a script.

The way Unity allows the user to view the 3D world is through the use of a camera object, whose X axis, Y axis, Z axis and angle can be statically set or changed by using a script. When an application is run, the window the application displays is the same view as the camera. In order to make the application compatible with Google Cardboard, instead of using the camera provided by Unity, it needs to use the camera provided in the Cardboard SDK package. This camera can be found by looking at the **Assets** folder, then clicking on the **Cardboard** folder, and then the **PreFab** folder to find the prefabricated cameras. Choose the **Cardboard Main** camera and drag it into the assets hierarchy window. This camera differs from the Unity once because they provide a stereoscopic view compatible with the Google Cardboard. This camera gives the application a sense of depth and makes the application more immersive.

The IDEALvr application is being developed for Android phones because they are the only device that supports Google Cardboard. In order to build and place the application on a device, go to **File** in the menu bar, then **Build Settings**, and select the **Android** format. Under **Player Settings**, set the bundle identifier to something appropriate, in our case **com.idealvr.WordCloud**. Click **Build**. When the application builds, it will create an Android Application Package (APK) file. During the build process, Unity will automatically look for any attached devices or virtual machines. An android phone or VM with an API level of at least 20 is necessary to run the application. Select where you want to place the generated APK file and it will automatically launch the application on the selected device. The user need place the android phone into the hardware component Google Cardboard in order to visualize the application appropriately.

If the IDEALvr application is ported to a web application, we will change the camera and use the original camera provided by Unity. Code for taking user input through the keyboard, ideally the arrow keys, will be implemented so the user can have a range of view similar to the Cardboard implementation. Instead of building an Android format, we will choose a web application.

2.2.2. Generating Word Cloud

An important aspect of the IDEALvr project is generating the word cloud of categories, keywords and tweets which will surround the user camera. In Unity, this word cloud will be represented by a single object with a script attached to it.

In the Unity scene created in section 2.2.1, go to **GameObject** in the menu bar and create a new empty object named WordCloud. Select this object on the scene window and in the properties window, attach a new script called FormWordCloud.

This script takes two parameters: the child object which is the object that will be displayed in the word cloud and the size (radius) of the word cloud. The child object for this word cloud is going to be a 3D Text object. Under the **GameObject** option, select **3D Object** and then create a **3D Text** object. Right click on the 3D Text object in the object hierarchy window and create a prefab of it. A prefab (prefabricated) is an object with already set properties. This will allow the script to create several of these prefab 3D Text objects rather than instantiating new ones. Delete the 3D Text object that was placed into the scene because it is no longer necessary. Click on the WordCloud object and set the child parameter as the **Word Cloud** object and the **size** parameter as 120.

The script consists of a Start method which is called when the scene is generated and the WordCloud object is created. The Start method calls a method which retrieves the JSON file of the words to populate the cloud with, as described in section 2.2.3. The Start method then calls a method to parse through the JSON file, turning each JSON entry into a Phrase object which holds the text and occurrences of each entry. This method also aggregates the occurrence counts and creates a total value of the occurrences which will be used to determine the size of the text. After these phrase objects are made and added to a List data structure, the Start method calls one last method that generates the positioning of the phrases in the scene. This method uses an algorithm developed by Andrew Sage[4]. This algorithm generates a list of points evenly distributed in a sphere shape. The points hold the X, Y and Z coordinates which the phrases will be placed at in the scene. For every point, we create a new child object which in this case is the 3D Text object and attach it to the scene.

This script generates the word cloud and displays it in the Unity world. Interaction with the world cloud is discussed in section 2.2.4. When a user clicks on any of the word cloud's child objects, we will change the JSON file that the world cloud reads from and regenerate the sphere.

After the implementation of the necessary aspects of the word cloud, testing was performed with different collections in order to make sure that the word cloud was readable by users. It was noted that the word cloud for categories with a large amount of collections will create a jumbled mess with too many words, too close to the user. It was impossible to read and understand what was going on. The script attached to the word cloud object in Unity takes in a size parameter. This size is the radius of the word cloud in relative pixel sizes defined by Unity. A technique of trial and error was used to determine a word cloud size which would fix this problem while keeping all other word clouds readable. A radius size of 120 was the optimal solution to keep words from being generated on top of each other and too close together.

The second problem related to the scale of the word cloud was when certain collections or keywords had a massive amount of occurrences. This would make some words simply too large for the user to make out what they are. Conversely, the same problem occurs for objects that have too little occurrences. They become too small to detect. A limit was added when forming the scale of these words according to their occurrences. If a word appeared in the top 20 percentile of the list of words ordered by occurrences, then the word size was decreased. If it appeared in the bottom 20 percentile, the word size was increased.

2.2.3. Populating Words from JSON file

In integrating the JSON files into the project, the first step is to assemble the formatted IDEAL data into a JSON format specific to our application. There are multiple ways to do this, but for us that is done in the word cloud structure and generated from the IDEAL data. Our JSON reads in the following format:

```
private string jsonString = "[{\"term\":\"Japan\", \"occurrences\":504}...]";
```

The objects start and end with '{' and '}'. Between them, a number of string value pairs can reside. String and value is separated by a ':' and if there are more than one string value pairs, they are separated by ','. The syntax for our JSON follows the general syntax of { string : value, ...} where "term" and "occurrences" are our string types. These are the only two fields that are needed for this step in the data collection per object. The term for every object identifies the data that will be displayed visually and the occurrences identifies the frequency of the term being used in the metadata. Together, the term displays the specific word to represent the data and the occurrences determines the size of the text object displaying the word for the term value of the object.

2.2.4. Data Structures

The average file is 10-60 MB. There are 8 different categories tagged in the IDEAL data so there would be 8 directories that all collections were sorted into. Being that the individual collections do not have a category tag, they must manually be sorted into their corresponding category directory. With over 1000 collections, it could take a few hours to manually sort all of them into the correct category directory.

The data structures used in this project consist of directories, files, and JSON strings. Categories are represented by directories, collections are represented by files, and the individual tweets are represented by JSON objects.

2.2.5. Implementing User Interaction with Cloud

The next step in the process was to develop a way for users to interact with the interface. We wanted the user to be able to select an option on every view to be able to advance to the next word cloud while specifying the category more. Working with a VR headset means that the user does not have interaction tools besides their own movement to help them guide. This means we will be capturing the line of sight for the user to be able to interact with the cloud. The process for interacting with the cloud is coded in the Text Selector script.

To accomplish this, we started off by creating an empty object for the scene to become a controller for the selected object. What is important in this step is identifying objects in the word cloud to be selected. We needed to create a script to be able to only select words in the cloud and ignore other objects that are a part of the scene created in Unity. To accomplish this, we set a global object that is updated per frame in the rendering of the view, which is rendered at 30 frames per second.

When we start to update the selection object, the first thing that is instructed in the controller is to capture the motion of the camera. After recognizing the position of the camera, the next step in the algorithm is to develop a way to capture the word selection. The first step in this process is to recognize the object type “Word” and then set it to a variable if the word is hit with the camera focus. Once the camera focus captures the word, there is a check for a value to make sure we have a target object. Following that we transfer the object data to a TextMesh object that will carry the phrase inside the object in the virtual reality. Once the word is captured, another check for the word is done before we move on to act on the object. Once we recheck the object’s location and phrase, we set the phrase coloring to highlighted. Outside our action case we also developed a way to remove the selection. This process is controlled by first checking if there is a current highlighted object and if there is, we follow that check with another color setting on the current object. This setting just undoes the highlighted effect on the current object to deselect it and return it to its original state to match the rest of the unselected word cloud.

In a visual summary, we start off by moving the camera to where the user is facing. One the camera recognizes the center of the frame, the object in the center of the frame is changed to a highlighted color. This color as of now in implementation is black in contrast to the default color of white for the word cloud words. When the user moves away from the selection, the word returns to its default color. This process dynamically lets the user make a selection with just the movement and shift of their view.

The next step was to implement the ability for the users to be able to interact with the interface by acting on their selection from the view. For the purposes of our presentation of the application, we use Google Cardboard to implement user interaction. This provides us with one hardware interaction that is available from the user endpoint, which is the select click trigger on the Google Cardboard.

We include a new script data field in the Text Selector script and set it to the Word Cloud object through the Unity application window. To capture the button trigger events, we needed to provide a listener algorithm to the application. Generally the listener would have to only react when a certain object is in focus. When the focus object is captured and the button is triggered simultaneously, then we can provide the user with a reaction in the visualization. The basic functionality for this capture is the same as the capture for the word selection, but also follows another reaction set where we capture the trigger by implementing the method GetMouseButtonDown. Inside this listener, when the user selects the current object and hits the button, we capture the hit word as a new variable and send this to the script data field by calling a method on the script. This script reads the input given and reforms the word cloud to the collection view or the frequent words view.

The folder hierarchy and files are named in such a way that the script attached to the Word Cloud is able to determine if it is in the category, collection or frequent word cloud. If the cloud is in the category view, the field sent will be the name of the category. The folder hierarchy includes a folder with this name so the script reads for the input JSON file in this folder. When a collection is clicked in the collection view, the script knows what category it is currently in, so instead of reading from the input JSON file, the script reads from the frequent word file of that specific collection. If the data field sent was of the value “BACK”, the method will return the user to the previous view of the cloud. This shows us that we have the selection of the user using the button captured and will allow us to further enhance the application using this data. In implementing the use of the button here we have provided the user with a way to advance the view based on the word selection they make.

2.2.6. Processing Collection Hierarchy

To display the word cloud with categories, keywords, hashtags, etcetera, the IDEAL data had to be organized in a manner that the Unity portion would be able to interpret. The first step was to manually create folders for each category for the given data. Each collection, based on a hashtag or keyword, was then put in its respective categorical folder. With this organizational structure, finding the size of a category would consist of counting the amount of lines contained in each file of that category. The folder (or category) name would then be saved to a file with the amount of occurrences contained for that category. The same was true for an individual collection. The number of lines in a file could be counted then saved to a JSON string along with the name of the file (or collection).

The JSON objects in the original IDEAL collections include very large JSON objects like the one example below:

{"created_at":"Sat Nov 21 14:22:47 +0000 2015","id":668072068177637376,"id_str":"668072068177637376","text":"It would be difficult to overstate what he has meant to and done for my school - #ThanksFrank","source":"\u003ca href=\"http:\/\/twitter.com\/#!\/download\/ipad\" rel=\"nofollow\"\u003eTwitter for iPad\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":159333587,"id_str":"159333587","name":"Biff Gaut","screen_name":"purplebari","location":null,"url":null,"description":null,"protected":false,"verified":false,"followers_count":11,"friends_count":20,"listed_count":0,"favourites_count":78,"statuses_count":77,"created_at":"Fri Jun 25 02:53:31 +0000 2010","utc_offset":null,"time_zone":null,"geo_enabled":false,"lang":"en","contributors_enabled":false,"is_translator":false,"profile_background_color":"C0DEED","profile_background_image_url":"http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile_background_image_url_https":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile_background_tile":false,"profile_link_color":"0084B4","profile_sidebar_border_color":"C0DEED","profile_sidebar_fill_color":"DDEEF6","profile_text_color":"333333","profile_use_background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/3126060339\/be28920d11ba2cbbfa5711112dad2ee5_normal.png","profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/3126060339\/be28920d11ba2cbbfa5711112dad2ee5_normal.png","default_profile":true,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors":null,"is_quote_status":false,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[{"text":"ThanksFrank","indices":[81,93]}],"urls":[],"user_mentions":[],"symbols":[]},"favorited":false,"retweeted":false,"filter_level":"low","lang":"en","timestamp_ms":"1448115767889"}

The goal of the parser is to read through all of these collection files and create JSON files with only the necessary information for the application. This includes the terms to be displayed in the cloud and the number of occurrences. This helps save storage since the application is used on a phone with limited space available. Below is an example of the JSON file generated by the parser:

```
[{"term":"ThanksFrank", "occurrences":10151},{"term":"VeteransDay-shaun",
"occurrences":9029}]
```

This process could be done manually for each category and collection, but for a useful application this would be a painstaking, tedious task. A parser was created to solve this problem. The parser reads through the original data directory, then uses the processes previously described to create corresponding JSON files for each category or collection.

The parse program consisted of four functions: main, listCategories, parseCategories, and outputJSON. The main function first creates an ArrayList where the collection category names will be stored. Next, the file where the output JSON string will be stored is created and opened so that it can be written to later. Now the category ArrayList and the directory where the categories are stored are input as parameters to the listCategories function.

In the listCategories function, the input directory is looped through and each folder is added to the category ArrayList. It is critical that only folders in this directory are added, being that category names are the only information being extracted at this point in the process.

After retrieving all category names, the program returns to the main function to process each of these directories by looping through each category and inputting them as the parameter for the parseCategories function.

Inside the parseCategories function, the input directory (category) is explored to extract the contained files (collection records). First, a file is created that is named after the category and stored in the directory of the category. This is where all of the JSON strings for each collection in this category will be stored. Now each of the files in the directory are looped through and preprocessed to delete blank lines, unnecessary spaces, as well as strip off the beginning of directory names and filetypes. Also, the lines for each collection are counted then output with their respective collection name

(name of cleaned up file name) to the proper text file for storing collection information in the category.

Printing is accomplished using the outputJSON function. It takes a string and occurrence integer as parameters. It then outputs the correct corresponding JSON string to express the category, hashtag, or keyword along with the number of tweets contained. This function is used when looping through individual collection files as well as for entire categories.

Once the program has finished running there will be one file in the original project directory, as well as a file in each of the subfolders (categories). These files each contain the proper JSON strings to represent the information stored in the current level of the hierarchy.

2.2.7. Forming Category and Collection Display

When looking at the current state of the application, we have developed the categories and collections to be displayed in word clouds. This is done using our class object FormWordCloud and is updated to be populated with the category and collection JSON files that were created in the processing collection hierarchy step of our application. Once the processing is completed, the JSON files are ready before the application is run.

The first step in forming the first environment, which is the Category cloud, is grabbing the JSON objects from the generated files. This gets into a deeper level of JSON libraries we used to integrate the JSON files we created. On the top level we refer to the specific text files that contain our JSON strings inside the getJSON function to create associations that will later let us analyze that strings.

The next step is to parse the JSON holding all of the categories. This is completed inside the ProessWords fucntion. Here we take the key value pairs that were set up in the JSON strings and add them to an array object called randomisedPhrases. This will eventually lead to our organization of the categories in the word cloud. This is also a temporary solution and we will be working in the future to develop meaning behind the organization of the display of text to the user experience to produce a new way to analyze the data sets.

The next step is to actually display the the words in the word cloud. The next step is to call the function Sphere which will display the actual information grabbed from the JSON strings. In each key value pair there is a ‘term’ and a ‘occurences’ tag associated with each category. After calculating the positions of the words, the ‘term’ strings are set as objects and then placed at incremented locations around the user's view as to imitate a globe. This creates the final view for the user experience. This methodology is exactly mimicked for the Collection cloud display as well, but with just a different set of generated JSON strings to display the specific collection keywords instead of the category terms.

2.2.8. Frequent Word Analysis

There are an array of different open source data mining tools available. Using these tools the most frequent words will be analyzed in the specific collection that is selected after the selection of category. Common words such as “the” will be removed unless they are important to a specific recurring phrase.

The first step of this process will require all collection JSON files to be parsed to extract the tweets into a list or text file. Once the raw tweet data is obtained, the tweets for that specific collection will be processed to find the most common recurring words and phrases. These words will be sorted in decreasing order of occurrences. A JSON file with the first 22 words/phrases saved along with their number of occurrences will be displayed in a word cloud with size relative to the total occurrences.

Frequent word analysis attempts will have to be tested in many different configurations to determine the best set of parameters when analyzing tweets. Attributes such as the best words to ignore and whether words or phrases are more important will have to be determined to deliver the best user experience. Although many of these choices are quite subjective, there must exist an optimal combination where the organization of information brings out new realizations for a majority of users.

Implementation consisted of parsing each individual tweet while counting the lines in each collection. Each unique word is stored in a hash map along with its current number of occurrences. Once parsing is complete, the most frequent words are determined by finding words with the largest number of occurrences in the hash map. These files were then output to a frequent word file in a JSON format using the same methods as developing the word clouds for categories and collections.

The frequent word analysis during parsing still needs to be fixed. There is a bug that is causing inconsistent output, which is greatly exacerbated when stress testing using many collections. For example, sometimes a collection will produce a list of terms with very low occurrences after frequent word analysis. The problem appears to be random because a successive run will usually output the correct frequent word results.

2.2.9 Category and Collection Organized Display

The views of the categories and collections will ultimately differ based on what way the information is best portrayed to the user. This assists in creating a better way to analyze the data. Knowing that we have less categories than actual collections, we can create a system that is spatially aware of the amount of objects we will be displaying. We attempted to create a system where for the categories shown in the first view, would be mapped based on their occurrences and the relative size of the text for each category. For the Collection view, we wanted to create a way to display the largest collections centralized in the word cloud with decreasing occurrences collections are found further from the center.

In attempting to organize the category and collection views to be presented to the user, we attempted many different types of organization styles. When considering the categories view and the volume of objects that would be shown, we decided that it would be useful to have the categories organized by size rather than a randomized positioning. The categories would ultimately be organized so that the categories with the most collections would be seen at the centralized viewpoint of the user. This is to provide the user with a way to understand where they can retrieve the most information quantitatively.

We also wanted a similar technique for the collections view, where the collections would be organized by how populated they are by the amount of occurrences. The view would consolidate the most occurred words in the center of the user's view. Following those text object for the collections, the rest of the collections would respectively surround the larger collections. Instead of a circular cloud formation, we also wanted to spread the text objects in a more horizontal formation. This lets the user be able to view more words horizontally without missing any objects over or under them. This would be done by setting a threshold.

In attempting to change the views to these formats, we attempted to create calculations to apply on iterations of the JSON objects for each word object. This, however became a problem when trying to apply a position based on a sub-property of each JSON and a relative position of the user. The position of the words could be set statically once, but then as the user moves, the position of the words also change and this causes a problem when the user wants to select an object that does not start directly in front of them.

2.2.10 Porting Application to Web

A web application version of the IDEALvr project was suggested midway through the project in order to account for users who do not own a smartphone or do not have a Google Cardboard available. Unity provides the ability to download the project as a WebGL web based application. Research was conducted on how to build WebGL projects.

In order to run the application as a web platform there are certain changes that must be done to the Unity application to support interaction without the Google Cardboard. Open the ideal-vr project folder in the Unity editor. Delete the Cardboard Main camera object located in the left hand side of the editor. Select **GameObject** and then **Camera** to add a normal camera object. Select this **Camera** and change the **x** field to 0, **y** field to 0, and **z** field to -10. Then click on **File**, then **Build Settings**, select **WebGL** to build a web based application. If **WebGL** does not appear, search Unity for a guide on how to install WebGL support for Unity. Otherwise, select **Build and Run**. Unity will ask you to select an empty folder to place the built files. Select a folder and press OK. Unity will automatically build the application and open a localhost in the user’s default browser. Unity will automatically convert head movements to mouse dragging and user interaction to mouse clicking. There are several known issues with the web application build for the IDEAlvr application which makes a web application unusable.

The current build runs very slowly, regardless of what option is selected. Once Unity builds the application, it automatically opens it in a Localhost server. The application takes even longer to load, and clicking on it while it is loading will cause an exception to occur, forcing the user to have to reload the page. After the application loads, it begins the IDEALvr application, however all that appears in the developer build is a debugging window with Null Reference exceptions. Research on debugging Unity WebGL applications lead to having to run Unity debugging code on all of the scripts developed. The team’s inexperience with Unity limited the ability for debugging the IDEALvr application in this situation. A proper, working web application build was not accomplished at this time. Possible solutions are discussed in section 3.3.

2.3. User Workflow

Figure 5 below shows the workflow process of the user and how they would interact with our application. The rectangles represent the state of the application and the arrows represent user interaction.

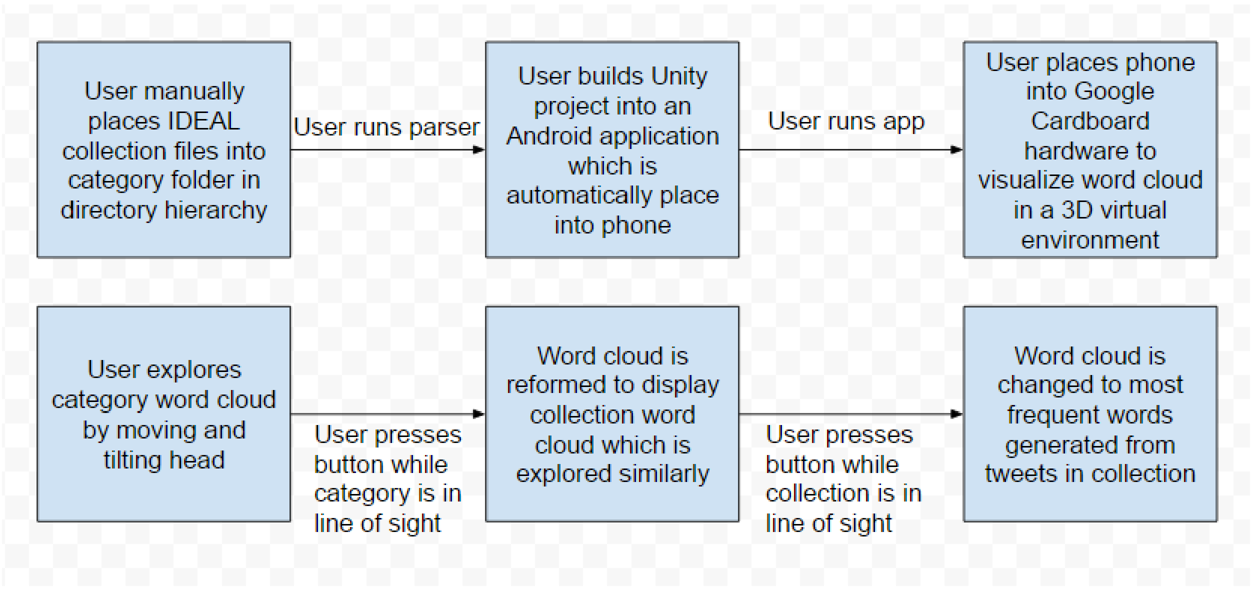
User manually places IDEAL collection files into category folder in directory hierarchy
User runs parser
User builds Unity project into an Android application which is automatically place into phone
User runs app
User places phone into Google Cardboard hardware to visualize word cloud in a 3D virtual environment
User explores category word cloud by moving and tilting head
User presses button while category is in line of sight
Word cloud is reformed to display collection word cloud which is explored similarly
User presses button while collection is in line of sight
Word cloud is changed to most frequent words generated from tweets in collection

Figure 5. Workflow model of entire IDEALvr application

3. Lessons Learned

3.1. Timeline

The timeline for the development of the IDEALvr application was created with the goal of having a deliverable application by the end of the semester. Several problems and reworking of the application made it difficult to stay on schedule. These problems are further discussed in section 3.2 and the reworked solutions and fixes are discussed in section 3.3. Every Monday, we met as a team with our client in order to discuss the current progress, planned work for the week and any insights and concerns our client had about the application. As a team, we communicated over messaging tools to make sure everyone was working and doing their designated tasks for the week. A comprehensive schedule of the work done on the application is located below.

Table 2. Timeline of work accomplished for IDEALvr project

Work Schedule	
2/1 - 2/8	Set up a Unity project and figure out how it works. Discussions on Unity techniques and ideas for application. Research on existing virtual reality visualization techniques.
2/8-2/15	Implemented ability to generate and display a static JSON string in a format parsable by our current Unity application.

2/15-2/22	Implemented ability to display text in spherical word cloud on Unity engine. Incorporate the parsed and generated JSON strings from last week and display them in the word cloud.
2/22-2/29	Implemented ability to move head around and explore the virtual cloud. Detect center of user vision and change color of word in order to indicate selection. Minimal interaction with cloud to allow user to click a word and receive a visual response.
3/14-3/4	Build on previous week implementation. Added logic for changing view when clicking on a category or collection keyword. Added a back button on every cloud to allow for back and forth interaction.
4/4-4/11	Created a parser to look at collections and develop a JSON file of frequent words. Implemented ability to show frequent words in word cloud after clicking on certain keyword. Attempted different techniques for displaying categories and keywords in a non-random way.
4/11-4/18	Conducted user testing to determine heuristic problems with word cloud size, text fonts and usability issues. Stress testing for crashes and other debugging tools.

A table of team members and responsibilities is located in section 4.1.

3.2. Problems and Solutions

Compiled in this section are all of the problems that occurred while working on the application and solutions or proposed solutions for the problems.

Table 3. Problems and solutions in the IDEALvr application

Problem	Solution
Could not save necessary JSON files onto local storage in the Android device.	Unity will look for a folder called StreamingAssets in the Asset directory. Anything in this folder will be sent to the Android device and keep the state relatively the same. Allowing us to read files from the Android rather than a computer or server.
Implementing an additional date cloud to further filter the collections by temporal relevancy did not increase cognitive analysis.	We did not have sufficient time to create another step of the parser to analyze the dates of a tweet and place them into separate folders/files. Since most of the tweets collected are also retweets, we thought it would not affect cognitive

	functionality if we separated the collections further down by date. The date cloud was scrapped.
Tweet visualization in virtual reality is not efficient or pleasant.	After user testing and analyzing feedback, it was discovered that reading an entire tweet in virtual reality proved to be difficult. It was also discovered that having to read tweets in their entirety would not help to conduct analysis on the tweet collection. Instead, we decided to develop a world cloud of the most frequent words. Therefore, the user would not need to read too much in an uncomfortable way to be able to conduct social experiments or analysis on the tweet collections.
Issues with getting correct and repeatable results with the frequent word analysis.	There are some issues with getting correct and repeatable results with the frequent word analysis. The frequent word analysis code will sometimes produce results that are less than favorable and sometimes will not produce the same result in separate runs. The issue is likely file writers and readers not being closed correctly.
Problems with porting to web application.	There are many problems with porting the application to a WebGL application. It is likely not able to locate the collection files. This requires debugging skills with Unity that none of the current team members have.
Problems with the calculations of several text objects in specific positions when iterating entire array of objects with size properties.	There are problems with the calculations of several text objects in specific positions when iterating entire array of objects with size properties when developing an ordered view of the word cloud for the category and collection view. Further debugging is necessary, maybe even a new positioning algorithm.
Difficulties arranging a specific position for each object in relation to property as the position index of objects is relative to the user.	There are difficulties arranging a specific position for each object in relation to property as the position index of objects is relative to the user. Further testing and implementation is required.

4. Future Work

With IDEALvr, we have achieved the basic functionality of what we set out to accomplish and there is still a future with the development that we can look forward to. We also faced a lot of limitations based on the time constraints we had and the allotted time to research the possibilities of how different ways of data visualization could benefit users. The future work will also consist of fixing the problematic situations we faced and overcoming them by finding solutions.

For this project, we did not visualize tweets ultimately, but in the future, it would be interesting to figure out a method that would make the visualization of the tweets beneficial to critical analysis of the data. The use of other types of metadata from the tweets would also be added into the future implementation of this project, including using other fields in the data objects other than the occurrences of each category and keyword. This would provide new ways to gain a deeper understanding of the data from analysis.

In the future, we would also like to be able to port the software to any device, whether it be mobile or on the web. This makes the application more accessible to a larger, more diverse pool of users. This is essential in providing a product that anyone can use. If not completely duplicable into other formats, it would still be useful to recreate in a different format for each platform.

One other implementation for the future would be to create new ways to organize the data for each view starting from the categories to the collections. This would provide a distinct way for each view to be complemented by the best way to view the data sets and also provide a distinct differentiations between the pages so as to not confuse the user while they advance through the application.

The frequent word analysis during parsing still needs to be fixed. There is a bug that is causing inconsistent output for only one of the collections every run. Once that problem is solved, there will need to be testing done on the application's use with a large set of collections. Further work and debugging needs to be done so that a web application build is fully functional. In its current state, the web application build does not support any visualization or user interaction.

The last step of refinement, after debugging, would consist of furthering user experience while using the application. The current application is simple to use; however, intuitiveness of the current design could be improved by utilizing modern researched methods of manipulating virtual reality interfaces. There is also a possibility for the option to choose between the top words or the top phrases after analyzing a collection. This will only be implemented if there exists an obvious need for both perspectives when viewing the word analysis cloud.

5. Acknowledgements

5.1. Team

The people working on this project will be dedicating their time on a version controlled application developed in versions until the end of the project. There are three main team members who will be developing IDEALvr.

Table 4. IDEALvr group members and responsibilities

NAME	RESPONSIBILITY
Luciano Biondi	Retrieving data from the IDEAL system and determining how to structure and display the data. Algorithms for interacting with visualized data.
Dagmawi Yeshiwas	Virtual reality techniques and software for maximizing the user base of the application
Omavi Walker	Data visualization and organization including how to color, the shapes, locations of objects and other factors including design.

5.2. Client

The client for IDEALvr is Virginia Tech researcher Mohamed Magdy. He is currently a computer science graduate student supervised by Edward A. Fox and Riham Mansour. He played a key role in the development of IDEAL and also controls the metadata that we will be integrating into the IDEALvr system. Mohamed provides the exported JSON data that will be viewed in our application. He also advises the team on the development of the application and provides constant resources in using the IDEAL database. The IDEAL project is supported by NSF grant #1319578.

5.3. Support Developers

One of the main supports that helped IDEALvr initiate development was the skeleton framework provided by Andrew Sage. Andrew Sage is an iOS and Android developer researching experimental virtual reality technologies[4]. He provides a tutorial at his personal blog extensively detailing how to create a virtual reality word cloud in Unity.

6. References

1. IDEAL database. Web. 24 March 2016. http://hadoop.dlib.vt.edu/

2. Digital Library Research Laboratory at Virginia Tech. Web. 1 May 2016. http://www.dlib.vt.edu/

3. Laha, Bireswar. "Effects Of Immersion On Visual Analysis Of Volume Data." *IEEE Transactions On Visualization & Computer Graphics* 18.4 (2012): 597-606.

4. Sage, Andrew. "A VR Word Cloud in Unity." 07 Oct. 2015. Web. 29 Feb. 2016. <https://medium.com/@SymboticaAndrew/a-vr-word-cloud-in-unity-f7cb8cf17b6b#.nrys96obv>.