I Want to Believe: A Visualization of UFO Siting Reports

Theodore Smith[1] Hailey Guillou[2]

1. Bioinformatics, University of British Columbia 2. Computer Science, University of British Columbia

Abstract

In this work we present an interactive visualization tool for examining UFO reports. The high level goal is to support 1) a geographic summarization of UFO sightings and 2) a linguistic summarization of these reports. By using manipulation tools, such as filter and zoom, we show how our visualization can aid curious individuals in gaining an insightful summary of the thousands of UFO reports available. Corresponding scenarios are provided for each task and an in depth review of potential future work is outlined.

INTRODUCTION

Have you ever looked into the night sky and seen something unusual? Perhaps it was a star or an airplane, but what if it was actually something from another planet? While some people may see something suspicious in the sky and keep going on with their lives, others feel the need to report this strange event to the appropriate authorities.

The National UFO Reporting Center (NUFORC) was founded in 1974 and since then its primary function has been to receive, record, corroborate, and document reports from individuals who have witnessed unusual, possibly UFO-related events. [1] NUFORC has had a 24-hour hotline available since 1974 and in the past 20 years has expanded to receiving reports via fax, email, and the organization's webform. These reports are stored in a publicly available database and are indexed by date of sighting, shape, state, and date posted. The reports range from serious reports to obvious hoaxes and while mostly reported from within the United States, there are reports of sightings throughout the world.

Initially when looking through the data, we thought it would be interesting to plot the reports geographically based on population density with the intention of searching for trends over time in terms of location, shape, and duration. With some investigation we found that current visualizations tend to address the question of who the people making reports are and where they are from. Trends in shape over time was also a popular visualization, as well as what time of day and what month have the most sightings. Upon gathering this prior work, we decided that while mapping the reports is still of interest, we also wanted to do some sort of textual analysis of the summaries of the reports in aggregate. We came to this conclusion after spending some time with the raw data and seeing that the text summaries often paint a more vivid picture of the encounter than the quantitative and categorical data possibly could.

What we propose in this work is a dual-view interaction tool which provides a geographic view that plots the reports on an interactive map and a textual summary of the aggregated reports. We use filters to reduce the number of items on the map based on shape, date range, time of day, and duration. The map view can be zoomed in to limit the visible data points. When there are multiple markers in a small area in the map, they are clustered together and encoded by hue to indicate the density of the markers within an area. Once a view on the map has been established, textual analysis can be performed on the visible reports using SentenTree, a visualization tool originally intended for text-based social media posts (tweets, etc.). The appearance of each SentenTree report is similar to a word cloud with the additional benefit of preserving an aggregated visual representation of sentence structure.

DATA AND TASK ABSTRACTIONS

Data description

Data were drawn from a scrubbed version of the NUFORC UFO sighting reports data set found on the machine learning and data science platform Kaggle [4]. This data set represents 80,327 reports from the original data set of 88,875 reports, with cases dropped due to missing location as well as missing or erroneous time. This data set also features a standardized form of the duration of each sighting, with time reported in seconds rather than in the mixed formats found in the original data set. Geographical coordinates were generated for each report using geolocation based on the city, state, and country fields in the original data. Comments were retained in full-text form without editing.

These data were subjected to a second round of cleaning in which reports containing missing or corrupted information were dropped. This was accomplished using Python's pandas module by means of importing directly from the original scrubbed CSV file to a pandas dataframe in order to efficiently check the number of columns in each row, as well as the contents of each column. An additional 2303 cases were dropped according to these conditions, resulting in a final data set of 78,024 reports.

Two variants of geographical encoding of the data were employed, both of which were drawn directly from the coordinates provided in the data set. At low zoom levels, where a large number of reports were visible within the map frame, individual reports were clustered using Leaflet's native MarkerCluster plug-in. Using this encoding, reports were grouped into color-coded point marks. These marks are uniform in size and vary in hue based upon the number of points contained within each cluster. Small clusters are represented with green marks,

medium sized clusters are represented with yellow marks, and large clusters are represented with orange marks. Due to the smaller number of visible reports at higher zoom levels, these clusters inherently become smaller as one zooms in on the map. Resultantly, clusters are regenerated at each zoom level and the relative cluster sizes are recalculated, producing a new distribution of cluster colors. At fine zoom levels, cluster markers are replaced with individual point markers for each report. These point markers resemble a map pin with the tip of the marker extending from the latitude / longitude coordinate pair for each report.

Upon clicking a point marker, a detail view is generated for the clicked report. This field contains text describing the date, shape, duration and comment associated with the specific report. These values were drawn from the data set and formatted using HTML.

SentenTree results are presented using the tool's default configuration with input drawn from the comment field of reports featuring coordinates which fall within the geographical bounds of the map frame at the time of SentenTree generation. SentenTree encodes the frequency of each word in the input data using size, and indicates aggregated sentence structure using line marks between the words.

Our tool also features a control panel utilized to specify filter settings. Each filter category corresponds to a field within the data. Specifically, we incorporated filter categories for shape, date, time, and duration. Due to the categorical nature of the shape field in the data set, we elected to constrain user input using checkboxes corresponding to the shapes described by NUFORC. Date is defined in terms of year, month, and day, while time is defined in hours and minutes. Both of these filter settings correspond to the datetime field in the data set. Duration is defined in seconds and corresponds to the standardized duration field in the data set. An additional filter field was provided for keywords and phrases, which corresponds to the text-based contents of the data set's comment field.

Task description

While much can be learned about the quantitative nature of NUFORC sighting reports through simple descriptives such as report density based on date, time, location, and shape, such an analysis overlooks the subjective experiences of the reporters. UFO sightings are neither consistent nor easily explained and, consequently, no two reports should be considered perfectly comparable. Our main goal in producing this visualization was to provide greater access to the quantitative properties of each report as a means for uncovering hidden features and commonalities in the experiences of the reporters. We view this as a two-fold task, wherein the data set can be intuitively explored using the more consistent features of each report such as time and location, and examined in greater detail using the linguistic features of the reports.

Our approach revolves around the separate, but related, tasks of representing the data in a spatiotemporal format as well as representing the linguistic relationships between isolated report descriptions. The data are geographically mapped based upon the locations specified in the reports. Users control the reports which are included in the linguistic analysis by means of navigating around the map using zooming and panning. Additionally, the user has control over the intervals of time and duration they are interested in viewing through the filter settings available in the control panel of the tool. Users also have control over the qualitative properties of filtered reports by means of a categorical shape filter as well as a keyword filter. This enables users to specifically target subjective types of UFO sightings based on the personal experiences of the reporters. The ultimate function of this component of our tool is to allow users to refine the complete data set of reports to a smaller, more targeted subset of reports which exhibit an enrichment of linguistic features of interest via an intuitive, geographically-driven interface.

Due to the large number of reports and highly variable comments, we deemed manual comparison of the subjective qualities of each report to be arduous and inefficient. Based on this observation, we were motivated to produce a visualization which provided an aggregated view of the comments. To this end we incorporated a SentenTree frame, which aggregates the targeted reports and generates a summary of their linguistic content. Moreover, the output of the SentenTree algorithm can be cloned for each combination of filters settings and geographic window to allow direct comparison to the output generated from another search. These searches can be performed iteratively by using the SentenTree output to manually specify a new filter configuration.

-

RELATED WORK

UFO Data

The most sophisticated prior work we uncovered in the visualization of UFO sighting reports is likely the infographic provided by John Nelson of IDV Solutions [7]. This post includes a variety of visualizations of UFO sighting reports, each intended to convey a different property of the data. The first choropleth map shows a simple ratio of sightings per capita. The second map shows a bivariate mapping of sightings in the color dimension (dark slate for low-sightings and bright green for high-sightings) and population density in the opacity dimension (denser populations are more transparent). Additionally, trends in reported shapes are provided in the form of line graphs indicating prevalence over time. While this presentation is informative at the descriptive level, it lacks interactivity and fails to incorporate comments.

Another visualization, found on metrocosm.com [8], breaks down the data set by the number of witnesses for each UFO sighting. This was accomplished by comparing the locations and dates of the sightings, however it does not incorporate comments when determining whether multiple coincidental reports truly correspond to the same event. This visualization provides less summarial information than Nelson's infographic, but compensates with interactivity and the ability to observe individual sightings along with associated comments.

Linguistic Analysis

Word clouds are likely the most commonly known text visualization. They began as "tag clouds" on websites that would highlight the most popular tags from posts on the website. The visualization grew in popularity as it began to be used in other text documents. Since it no longer just applied to tags, "word cloud" became the commonly known term. A common variation of this technique is Wordle [3], which automatically generates layouts that are aesthetically pleasing with words displayed horizontally and vertically. The Wordle model was an interesting starting point for what we wanted to analyze, but the word cloud itself did not provide enough context about what exactly is in the summaries.

On that basis we elected to utilize SentenTree. SentenTree was originally developed for the analysis of Twitter data, and represents an improvement over the word cloud approach in that it maintains a visual representation of sentence structure. In doing so, SentenTree provides a sense not only of commonly occurring words within a data set, but also provides an impression of the meaning of those words by arranging them in an order driven by the semantics of the input data.

Solution

Table 1.

What: Data	Table. Items: UFO sighting reports. Attributes: date/time of sighting, shape (categorical), duration (quantitative), city (categorical), state(categorical), summary, date posted
What: Derived	Latitude and longitude coordinates, duration normalized to seconds
Why: Tasks	Explore trends in the textual summaries of UFO reports
How: Encode	Geographic view with hue encoding for number of reports in a cluster area; directed node-link diagram with size encoding for the word count
How: Facet	Multi-form: overview-detail
How: Reduce	Filtering items using widgets and filters onto the map view
How: Manipulate	Select (individual points on the map for a detail view); geometric zooming on the map
Scale	Tens of thousands of items

All features of our tool were produced using JavaScript, HTML, and CSS. Mapping of reports was accomplish using Leaflet and MarkerCluster while linguistic reports were produced using SentenTree, all three of which are JavaScript libraries. Visible reports were controlled by means of specifying filter conditions and navigating the map, while the SentenTree output was directly linked to visible reports. In this fashion, we produced an interactive means for controlling linguistic analysis of the data set. Further exploration of the linguistic contents of the data set was facilitated by incorporating the ability to clone the SentenTree output after a search, enabling comparison of those results to a subsequent search.

Implementation

Geographical mapping was accomplished using Leaflet [5]. Leaflet is an open-source JavaScript mapping library, similar in some ways to Google Maps with the advantages of allowing manipulation of the underlying code as well as not being bound to a specific map projection. We considered a number of map projection options before ultimately selecting OpenStreetMap, primarily for aesthetic reasons. An additional consideration underlying this choice was the fact that OpenStreetMap was founded in 2006 and continues to have a strong user base. This inspires confidence that our tool will remain functional for the foreseeable future.

In order to plot the reports using Leaflet, the CSV data file was pre-processed using python and necessary fields including datetime, duration, shape, latitude, longitude, and comments were extracted. Python was also utilized to generate HTML for each report by formatting the date, time, duration, shape, and comment fields in a separate, encapsulated field. The final output format was an array of arrays, with each inner array representing the data for a single report. The outer array was exported to a JavaScript file, which was subsequently imported by the JavaScript code for the tool itself.

The main functional elements of the tool were programmed using JavaScript, wherein all reports were stored in a JavaScript variable on page load. During use, users specify a set of filter conditions. These conditions are set using the HTML fields found in the control panel of the tool. Upon submission of these conditions, a number of JavaScript functions are triggered which check the values of the filter fields and store the values in a set of arrays. These arrays are subsequently passed to the main logical component of the tool which sequentially compares the contents of each row in the data set to the conditions. If a row matches the filter conditions exactly, it is added to a separate array of Leaflet

markers which ultimately determines the results which will be plotted on the map. Markers are an integral data structure within Leaflet containing the coordinates of a record along with a “title” field. This field generates the detail view of the point and was tied in our algorithm to the HTML detail field in the data set produced using Python. If a row fails to match any condition, the other comparisons are skipped, the algorithm moves on to the next row in the data set, and the row is not added to the displayed output.

Once markers are generated from the filter-matching reports, the markers are added to a marker group using Leaflet’s MarkerCluster plug-in [6]. Again, this is a data structure defined within the plug-in which allows the markers to be clustered and subsequently appended to the main map.

Due to the large size of the NUFORC data set, simultaneously plotting all reports on a map proved too computationally intensive resulting in slow loading time and unstable performance. Moreover, even on successful loading of all points, we found that the plotted points were too congested to provide a useful view of the geographical distribution. Rather than generating an independent SVG marker for each report at low zoom levels, we instead chose to employ Leaflet’s MarkerCluster plug-in. This plug-in clusters individual markers into groups based on geographic proximity as measured in pixel distance on the map.

While we considered developing our own clustering algorithm, we found the performance of MarkerCluster to be suitable for our purposes and appreciated its integration into Leaflet’s core functionality, likely resulting from the fact that it was produced by the same developers. While MarkerCluster proved to be an effective solution for plotting the data, we found that load times were still high due to the large size of the data set. To offset this, we took advantage of an option built into the original code which allows for “chunked” loading of points with triggering of external code upon loading of each chunk. With this option enabled, we adapted code from an example provided by the Leaflet developers in order to generate a progress bar which was positioned over the map. This progress bar provides visual feedback to the user to indicate that the tool is not frozen during loading of a large set of markers.

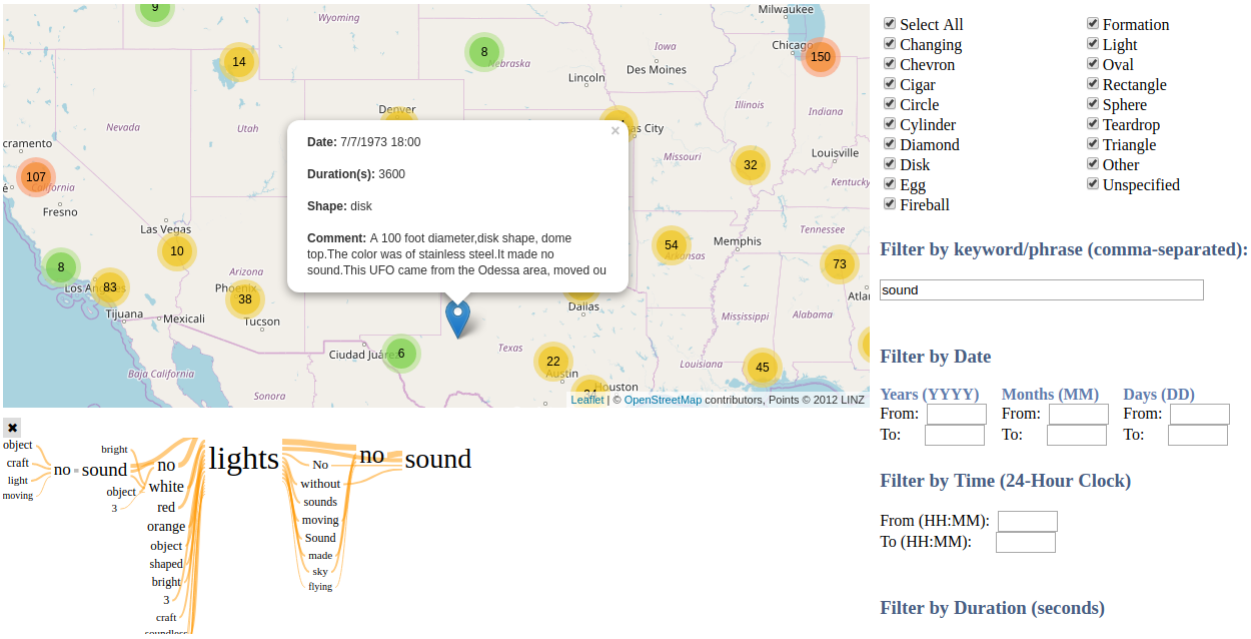
SentenTree was incorporated into our tool using the default settings of the original designers. In order to tie the SentenTree output to the reports visible on the map at any given time, the coordinates of the displayed results were compared to the coordinates of the north, east, south, and west bounds of the map frame. We appended results falling within those bounds to an array containing JavaScript objects with sequentially generated IDs, a SentenTree weight of 1, and the comment associated with each report. In order to properly parse the comment field, an intermediate step was performed in which the comments were passed through an HTML text field to convert HTML entities and formatting to plain text.

Formatting and styling were accomplished using HTML and CSS. The date selection component of the control panel was drawn from a third-party Bootstrap tool (CITE). All other HTML and CSS were produced by the authors.

Task	Hayley	Theodore
Data collection	0%	100%
Initial linguistic analysis implementation	100%	0%
Initial geospatial implementation	0%	100%
Data cleaning	50%	50%
Plotting data to map/SentenTree	0%	100%
Filters	25%	75%
Optimization of data load	0%	100%
Design - layout and styling	100%	0%
Slides	75%	25%
Final Report	50%	50%

Results

We find that this tool provides an intuitive means for exploring the data set while also revealing interesting subjective features of the reports. As an example, we were interested in investigating whether or not reports included any description of the sound heard during sightings. In order to address this, we began by filtering reports on the keyword “sound.” Our rationale was that some percentage of reports may be explained by people observing unfamiliar, yet conventional aircraft such as military jets, propeller-driven planes, or helicopters. One might expect that these aircraft would be audible, even if their sonic characteristics were strange enough to the reporter to warrant a report. We found a marked over-representation of the combination of the word “no,” with the word “sound,” suggesting that the UFOs sighted by reporters were often silent. Individual evaluation of a number of comments corroborated this observation, leading us to the conclusion that valuable information can be extracted from the output of the SentenTree frame.



With the exception of two stages of peer review during the development of this tool, we have not done any user studies or computational benchmarking. Feedback we received during this reviews was incorporated during our development and was mostly related to the responsiveness of the tool. This was addressed using the aforementioned MarkerCluster plug-in, as well as by applying filter changes in batches rather than individually. We have not followed up with these reviewers or tested the tool on a wider audience since finishing development of the tool.

Discussion and Future Work

A limitation we noted in our original design was that, short of manually taking a screenshot, SentenTree results from one search could not be directly compared to a subsequent search. By adding the option to clone SentenTree output, we allow direct comparison of multiple text analyses, yielding greater exploratory power.

However, even with this advancement over our initial design, the utility of the SentenTree output is limited by several factors. Firstly, the SentenTree output is not interactive, meaning that any search refinements based on the output must be performed manually. This is mainly by means of modifying the keyword and shape filters. Furthermore, using the map as a geographic filter is a somewhat crude method given that SentreeTree filtering can only be performed by moving the map. In future works, a valuable feature may be the incorporation of a selection tool which would enable users to select specific points or regions for SentenTree analysis. Finally, SentenTree frames do not encapsulate the filter conditions which produced the specific output. A future implementation may incorporate a sub-frame describing the filters and geographic region used to produce the SentenTree output to enhance continuity.

Once a SentenTree is created, the interaction stops. Currently it is simply a tool for pattern discovery over the entire dataset, but it would be useful to be directed to certain items from these patterns. For example, by hovering over a word in the sententree, the corresponding data points on the map could change colour or be highlighted or by selecting one or more words in the sententree, the corresponding reports could be listed in a separate view.

Each item in the SentenTree has an id, count, and text. The count is the weight of the text. In our implementation, we left all the

counts for the text comments to be 1 so that it was even. In a future iteration there could be weighting based on how many sightings are reported within a certain time window and within a certain radius. This may have some effect on the contextual ambiguity of the SentenTrees. Currently the context of sentences are preserved, but the result is vague. We do not know if this is from the data itself not being related in a meaningful enough way or if it is from our weighting strategy (evenly weighted), so this would be an interesting problem to explore in the future

At one point we considered using different marks for different reported shapes to add that as a visual indicator on the map. This would require extra processing of the data to reduce the number of categories as there are currently 18 different shapes. For example, circle, oval, egg, and sphere could possibly merge into one category and light and fireball could merge into another. The reduction of shape categories may connect more sightings as circle and oval could be a matter of perception based on location. This representation of shapes might be difficult with the current clustering methods; however, the marker clusters are a current limitation of our implementation and with more time we would have considered a more custom solution.

Conclusions

Although there is obvious room for improvement in this tool, we believe that we have produced a viable means for interactively exploring the linguistic contents of the NUFORC UFO sightings data set. Our tool provides greater access to trends in the comments associated with reports than individual assessment of the comments alone, and our interface provides an intuitive way to target reports matching a complex set of conditions in order to evaluate a particular feature of interest.

Bibliography

[1] http://nuforc.org/General.html

[2] Mengdie Hu, Krist Wongsuphasawat, and John Stasko, “Visualizing Social Media Content with SentenTree”, IEEE transactions on visualization and computer graphics 23.1 621-630. 2017

[3] F. B. Viegas, M. Wattenberg, and J. Feinberg. Participatory visualization with Wordle. IEEE Transactions on Visualization and Computer Graphics, 15(6):1137–1144, 2009.

[4] https://www.kaggle.com/NUFORC/ufo-sightings

[5] http://leafletjs.com/index.html

[6] https://github.com/Leaflet/Leaflet.markercluster

[7] http://uxblog.idvsolutions.com/2015/06/sightings.html

[8] http://metrocosm.com/ufo-sightings-map.html

[9] https://github.com/uxsolutions/bootstrap-datepicker