Department of Electrical and Computer Engineering

Part IV Research Project Report

10 September 2016

Grouping Similar Newspaper Articles

Author: Ruoyi (Zoe) Cai
Project Partner: Chanjun Park
Supervisor: Gill Dobbie

Project 72

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

Name: Ruoyi (Zoe) Cai

***Abstract*—An increasing number of newspapers are publishing articles online, and more users are accessing the news through websites. However, articles are commonly grouped under general topics such as 'National' and 'Business', which makes it inconvenient for readers wanting to read groups of articles that cover similar topics. This paper presents Clusterr, a web application that groups similar newspaper articles into clusters and visualises the clusters formed in an intuitive and user-friendly way. It presents the background of the application and the work done to implement the application, as well as evaluation and future work that could be done to further extend the application.**

I. Introduction

The purpose of this project is to group semantically similar documents into clusters and to present the clusters visually. Newspaper articles are used as the primary input dataset.

Clusterr is a web application that was developed to achieve this purpose. It clusters newspaper articles and visualises them intuitively. Clusterr allows the user to retrieve a specified number of newspaper articles within a date range. It then groups the articles with a common topic into the same clusters, and displays the clusters using interactive and intuitive visualisation.

Grouping semantically related documents first involves finding features which are used to distinguish documents in one group from those in another. In this case, these features should represent the semantic content of the newspaper article. Then, to measure the similarity of the documents, the features are used as inputs to a probabilistic data matching algorithm, which groups the documents based on their semantic similarity. Finally, the clusters are visualised so that the data could be interpreted easily.

This report explains the background of the project in sections 2 to 5, presents the research and implementation in sections 5 to 9, and concludes with the evaluation and future work in sections 10 and 11.

II. Motivation

Newspaper articles are usually grouped under categories like "Business" and "National", but never by topics over time. A news reader interested in a particular topic would have to manually sift through articles to find those of interest. The motivation behind this project is to automate this process, and present the resulting article clusters in a user-friendly and intuitive way.

III. Related work

A. Existing Clustering Applications — Search Engines

Document clustering is currently applied mainly in search engine tools to group results returned from a keyword search into categories. Some search engines that use document clustering are briefly described below.

1) SinglePoint

SinglePoint is an enterprise strategic research portal developed by Northern Lights Group, LLC. SinglePoint provides search and text analytics solutions to global companies, with typical applications in market research, competitive intelligence, market intelligence, technology intelligence, and customer intelligence. The search engine focuses on attributes commonly found in research reports, business documents, and industry news, and groups sources into business concepts, such as "Competitors", "Companies", and "IT Markets". Its sources include journals, financial analyst research papers, social media, business news, and eBooks. [1]

2) Yippy

Yippy is a metasearch engine developed by Vivísimo. It uses federated search to aggregate and sort results from other search engines into topics, so that users could further narrow down their search scope. [2]

3) Noggle

Noggle is a commercial desktop search and clustering engine that searches through local documents and cluster the returned results into groups. It is aimed at users searching through a high-volume document storage encountered in cloud and big data services. [3]

4) Carrot^2

Carrot^2 is an open-source search results clustering engine that fetches results from external sources such as Bing Search API and PubMed. It then organises the search results into categories and either displays them in a text-based 'Folders' view, or visualises the categories with the 'Circles' or 'FoamTree' visualisations. [4] It is implemented in Java and uses the Lingo algorithm, which is an algorithm developed by Stanislaw Osinski specifically for clustering search results. [5]

B. Google Trends

Google Trends is a web application that summarises the most popular search phrases within a particular time period and visualises that data. It allows users to see the current trends and what people care about during a certain time period. However, it does not show trending topics from the media's point of view, which this application focuses on.

IV. Focus and Scope

The focus of this project is the implementation of an application that makes use of existing algorithms. There is a lot of research being carried out on similarity measures and clustering algorithms, and many algorithms have been proposed. Although their effectiveness has been tested by their authors and creators, there is a limited number of real-world applications that implement the algorithms. Therefore, instead of creating and proposing more algorithm, the motivation is to make an application that implements and possibly combines the existing proposed algorithms, an application which could be useful to the general public as well as researchers and data analysts. Time constraint and the extent of our existing

knowledge on similarity measures and clustering concepts also contributed to this rationale.

In addition to clustering, a significant focus of this project is also on the visualisation of clusters in an intuitive and user-friendly way.

V. Problems to be Solved

- Find and retrieve newspaper articles for clustering
- Find reliable features that can be used to distinguish semantically dissimilar articles
- Filter features to find the ones that are the most significant to avoid overfitting
- Cluster the articles
- Evaluate whether the resulting clusters are logical and "good enough"
- Visualise the clusters intuitively
- Make a user-friendly web application that retrieves articles dynamically
- Deploy the web application

VI. Programming Language and Tools

Java was initially chosen as the main back-end programming language, as Java is the primary programming language used in courses at the University of Auckland and hence the most familiar to me. However, after research and some implementation, Python was found to be more suitable because of its extensive and mature machine learning tools, making implementation much simpler.

The following Python tools are currently used in the implementation.

A. NLTK

NLTK stands for Natural Language Toolkit. It is a suite of libraries and programs for natural language processing. It provides libraries for classification, tokenisation, stemming, tagging, parsing, as well as interfaces to over 50 corpora and lexical resources. [6]

B. Scikit-learn

Scikit-learn is an open-source machine learning library that provides tools for data mining and data analysis. It provides tools for classification, regression, clustering, dimensionality reduction, model selection, and pre-processing. [7]

C. NumPy

NumPy is a package for scientific computing with Python. It contains mathematical functions to operate on large multi-dimensional arrays and matrices. [8]

D. Flask

Flask is a micro web framework based on the Werkzeug toolkit and Jinja2 template engine. [9]

E. Gunicorn

Gunicorn is a Web Server Gateway Interface HTTP server compatible with Flask. It is simply implemented, light on server resources, and fairly speedy. [10]

The following libraries are used for the web application.

F. D3.js

D3.js is a JavaScript library that makes use of the SVG, HTML5, and CSS standards to produce dynamic, interactive data visualisations in web browsers. [11]

G. jQuery

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. [12]

H. Air Datepicker

Air Datepicker is a lightweight cross-browser jQuery datepicker. [13]

VII. Project Management

A. Methodology

The project began with general research into the different approaches to solving the problem. Everything that was found was added to a list for future research. From there onwards, it was a process of cycling between stages of implementation, validation, and research. Like cogs, each stage pushed the others forward. For example, new ideas would be generated during implementation which could be added to the list for research, and validation would improve implementation while debugging and tweaking the algorithm.

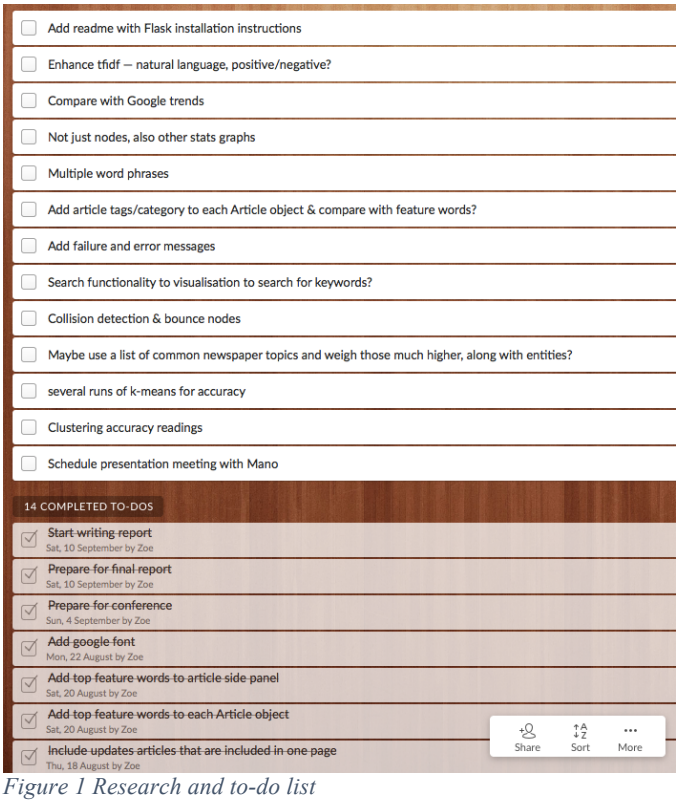

Figure 1 Research and to-do list

Git was used for version control, with GitHub storing the remote repository. A master branch was created to contain working code that has been reviewed and tested, and personal branches were created for new features, which are merged into master upon completion of each new feature.

After the first release, a pipeline was used which deploys a new version of the application with every pull request for review, an intermediate stage contained the deployment of the master branch, and finally the production stage contained the application displayed to the website.

A work log was kept for the work done throughout the year, which is maintained on GitHub Wiki. A total of 237.5 hours were logged as of 4 September 2016.

B. Milestones

March: Research into approaches to solving the problem
April: Begin implementation of pre-processing and document features extraction
May: Begin clustering algorithms research and implementation
June: Complete implementation for clustering visualisation
July: Begin web application
August: Complete web application. Test, validate, and fine-tune clustering algorithms

VIII. Research

A. Document Representation

In order to compare the similarity of documents, a numerical statistic must be derived from the document text to represent the document content. Tf-idf weights can be used for this. Tf-idf is a statistical measure used to evaluate the importance (weight) of a word to a document within a collection. It is calculated using term frequency and inverse document frequency. [14]

Term frequency is simply the number of times a term occurs in the chosen document, and represents how important the term is to the chosen document. The raw term frequency could be used, or it could be scaled by the total number of terms in the document. An example of a term frequency calculation is:

$$tf(t,d) = \frac{frequency\ of\ t}{total\ number\ of\ terms\ in\ d}$$

where it is the term and d is the document.

However, if term frequency were the only factor considered, all terms would be considered equally important, regardless of how much each term contributes to the distinctiveness of a chosen document. Therefore, inverse document frequency is introduced to attenuate the effect of common terms in the collection. It is calculated using the number of documents the term occurs in, and is often logarithmically scaled.

$$idf(t,D) = \log \frac{total\ number\ of\ documents\ in\ D}{number\ of\ documents\ where\ t\ occurs}$$

where t is the term and D is the collection of documents.
The term frequency and inverse document frequency would then be multiplied.

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

B. Named Entity Recognition

Named Entity Recognition (NER) could be used in addition to tf-idf weights to determine the most important terms in the document set. Named Entities are phrases that represent names of things such as people, companies, and locations. And Named Entity Recognition labels the Named Entities in the input text, allowing the phrases to be extracted as features. [15]
An example output could be:

[ORG U.N.] official [PER Ekeus] heads for [LOC Baghdad].

where "ORG" represents organization, "PER" represents person, and "LOC" represents location. [16]

C. Similarity Measures

To evaluate the extent of difference between documents, a numeric value must be calculated. A number of similarity measures have been created. The following are a few notable measures.

1) Euclidean Distance

Euclidean distance is the distance between two points. In this case, tf-idf weights are used to calculate the term vectors of two documents, d_a and d_b:

$$D_E(\overrightarrow{t_a}, \overrightarrow{t_b}) = (\sum_{t=1}^{m} |w_{t,a} - w_{t,b}|^2)^{\frac{1}{2}}$$

where the term set is $T = t_1, \dots, t_m$ and $w_{t,a} = tfidf(t, d_a, D)$

2) Cosine Similarity

Cosine similarity finds the normalized dot product of the two term vectors by calculating the cosine of the angle between the vectors.

$$\cos\theta(\overrightarrow{t_a}, \overrightarrow{t_b}) = \frac{\overrightarrow{t_a} \cdot \overrightarrow{t_b}}{|t_a| \times |\overrightarrow{t_b}|}$$

As in Euclidean Distance, tf-idf weights are used for the vector calculation.

3) Jaccard Similarity

Jaccard similarity is defined as the size of the intersection of sample sets divided by the size of the union of the sets. [17] In our case, it is:

$$Jaccard(\overrightarrow{t_a}, \overrightarrow{t_b}) = \frac{\overrightarrow{t_a} \cdot \overrightarrow{t_b}}{|t_a|^2 + |\overrightarrow{t_b}|^2 - \overrightarrow{t_a} \cdot \overrightarrow{t_b}}$$

Similarity measures for text document clustering have been compared by Anna Huang. The standard K-means algorithm was used to analyse the effectiveness of five similarity measures on seven text document datasets. [18]

D. Clustering Algorithms

A number of different clustering algorithms have been proposed. Two main types are partitional (unnested) and hierarchical (nested). K-means is a partitional clustering algorithm and is one of the oldest and most widely used clustering algorithms. [19] However, mini-batch k-means has been proposed to improve the scalability of k-means. Two different types of hierarchical clustering are also described below along with DBSCAN clustering.

1) K-means & Mini-batch K-means

K-means classifies a given document set into k clusters. It defines k centroids, one for each cluster, and attempts to minimise the residual sum of squares which represents how well the centroids represent their documents. The residual sum of squares is the squared distance of each vector from its centroid summed over all vectors. [14]

The K-means algorithm is composed of the following steps: [20]

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids reach convergence — no longer move.

A more efficient and scalable modification of k-means has been proposed called mini-batch k-means, which is able to converge to a near optimal value several orders of magnitude faster than the original full batch k-means. On large data sets, mini-batch k-means is several times faster, and "for small values of k, the mini-batch methods were able to produce near-best cluster centres for nearly a million documents in a fraction of a CPU second on a single ordinary 2.4 GHz machine, making real-time clustering practical for user-facing applications." [21]
Mini-batch k-means uses small random batches of examples of a fixed size obtained at each iteration, which are used to update the centroid positions until convergence. Each mini batch updates the centroid positions with a learning rate that decreases with the number of iterations, allowing the current mini batch values to be combined with the previous mini batch, so convergence can be detected when the clusters do not change in several consecutive iterations. [22]

2) Hierarchical Clustering

There are two kinds of hierarchical clustering techniques — agglomerative and divisive — which construct their hierarchies in opposite directions. Agglomerative techniques start with each document representing one cluster, and in each step merges two clusters, terminating once all clusters have been merged. In contrast, divisive techniques start with one cluster containing all documents, and splits up the documents in each step until each cluster contains one document. The process can be pictured as a tree, with agglomerative traversing from the leaf nodes to the root node, and divisive traversing from the root node to the leaf nodes.

Below is a basic agglomerative hierarchical clustering algorithm [19]

1. Compute the proximity matrix
2. Merge the closest two clusters
3. Update the proximity matrix to reflect the proximity between the new cluster and the original clusters
4. Repeat merge until only one cluster remains

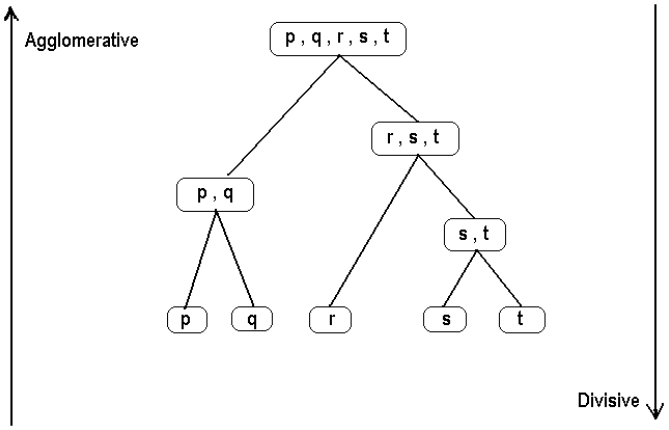

Figure 2 Agglomerative and Divisive [23]

Linkage criteria determine the metric used to calculate the proximity matrix used for the merge strategy. Ward linkage minimises the sum of squared deviations from points to centroids within all clusters, minimising the increase in total within-cluster variance at each step. [24] It is similar to the k-means metric. Maximum or complete linkage minimises the maximum distance between pairs of points in the two clusters being merged. And average linkage minimises the average distances between all pairs of points in the clusters being merged. [25]

3) DBSCAN

DBSCAN stands for Density-based spatial clustering of applications with noise. It is a density-based clustering algorithm that groups points that are packed close together in a space and marks the points that exist alone in low-density regions as outliers. Unlike k-means and agglomerative clustering, DBSCAN does not require the number of clusters as an input, instead requiring two other input parameters: Eps, the maximum radius of a neighbourhood, and MinPts, the minimum number of points in a neighbourhood.

The algorithm defines each point as either a core, border, or noise point. A core point is a point that has at least MinPts in its neighbourhood. A border point is a non-core point that has a core point in its neighbourhood. And a noise point is neither a core point nor a noise point, representing outliers.

The algorithm then assigns each core and border point to a cluster. It picks an unassigned core point in the dataset and performs a depth-first search starting at that point, adding each unexplored core point to the neighbourhood recursively. A tree structure is formed in the end which represents a single cluster. This is repeated until all core points are assigned to a cluster. [26]

E. Cluster Validation

The silhouette coefficient is a measure of how similar a document is to its own cluster compared to other clusters. [27] It is computed using the formula:

$$\frac{b_i - a_i}{\max(a_i, b_i)}$$

where i is a point in a cluster, a_i is the average distance of i to all other points in its own cluster, and b_i is the minimum average distance of i to all points in every other cluster.

The silhouette coefficient can range from -1 to 1, where a negative value corresponds to a case where the intra-cluster distance is higher than the inter-cluster distance and is hence undesirable.

The silhouette coefficient is used for cluster validation. As almost every clustering algorithm will find clusters in a dataset, even if the dataset has no natural cluster structure. The problem is not easily detected when clustering points in a high-dimensional space, therefore the silhouette coefficient is used to validate the existence of clusters in a dataset, as well as to determine the natural number of clusters in a dataset. [19]

F. Dimensionality Reduction

Principal component analysis (PCA) is a technique used to reduce dimensionality of a dataset while emphasising variation, highlighting the similarities and differences in the data. It compresses data without much loss of information, allowing high-dimensional data to be visualised. [28]

IX. Implementation

A. Pre-processing

Tokenisation parses text data into smaller units such as words and phrases as part of pre-processing. In this application, tokenisation is used to read all of the input documents and filter out anything that is not a word, such as punctuation and white spaces.

Regular expression is used to split the document into tokens of single words.

After the split, each word was initially converted into lowercase. However, since parts-of-speech tagging was added, the words now remain in their original form, so that named entities could be distinguished from verbs and adjectives which are later filtered out.

The stop words corpus in NLTK is used to check if any token is a stop word in English, which is removed from the tokens list.

Method pos-tag, the part-of-speech tagger in NLTK then tags each token with word classes according to the parts-of-speech tags used in the Penn Treebank Project [29], such as ‘NN’ for nouns. Only nouns, plural nouns, proper nouns, and plural proper nouns are kept in the list, as nouns were found to be more likely to contribute the distinctiveness of a document.

Nouns, verbs, adjectives, and adverbs were all experimented on, but nouns only were found to be the best features for distinguishing semantically dissimilar documents.

The Stanford NER [15] was also integrated at one point, where only named entities were used as feature terms. However, this also produced features that were not as accurate at distinguishing dissimilar documents as those produced by nouns.

The WordNetLemmatizer from NLTK is then used to lemmatise the tokens with the part-of-speech tags. Lemmatisation instead of stemming is used as lemmatisation creates lemmas with a context of the word meaning, whereas stemming simply uses common regexes to remove common word suffixes from a word regardless of whether the word is valid. For example, the lemmatiser would return “good” for the word “better”, and depending on whether the word “dove” refers to a noun or a verb, it word return “dove” or “dive” respectively.

B. Tf-idf

The dictionary of tokens is then converted into a matrix of tf-idf features. Each row represents a document, and each column represents a token. For example, if word j did not exist in document i, then tfidf_matrix(i,j) = 0.

The tf-idf vectoriser also limits the maximum number of words to four times the input cluster number. This was determined by trial-and-error, as four times the cluster number of feature words was found to be the optimum number that does not cause loss of too much information while minimizing the dimensionality of the final matrix for clustering.

The tf-idf weights of each word in each document are first calculated, then the words are filtered by ordering the words by the total sum of all tf-idf weights in all documents for that word. This sum represents the importance of a word to the document set, and allows us to then obtain the most important words to the document set, which are the words with the highest sums.

C. Clustering

Agglomerative clustering is used to cluster the matrix of tf-idf values. Complete linkage is used as its distance measure is the distance between two points (one from each cluster) that are the farthest away from each other, and therefore would lead to

higher-density clusters. Though complete linkage tends to cause the most significant "rich get richer" behaviour that leads to uneven cluster sizes [30], as the dataset consists of newspaper articles, even cluster sizes are not necessary.

For the range between two and the input cluster number, documents are clustered in iterations that test out the different cluster numbers. A silhouette coefficient is calculated for each iteration with a different number of clusters. Once the iterations are complete, the iteration that resulted in the highest silhouette coefficient is saved as the best number of clusters for agglomerative clustering, along with the clusters formed from the best cluster number.

Then mini-batch k-means is used to cluster the matrix of tf-idf values again. As in agglomerative clustering, the mini-batch k-means repeats clustering for each number of clusters in the range between two and the input cluster number, calculating the silhouette coefficient at each iteration. In the end, the cluster number with the highest silhouette coefficient is saved as the best number of clusters for mini-batch k-means, along with the clusters formed from the best cluster number.

The two saved silhouette coefficients from agglomerative and mini-batch k-means are then compared, and the clusters formed from with the clustering method that resulted in the highest silhouette coefficient are returned and used for visualisation.

DBSCAN was also implemented. However, as the documents in the dataset are not in obvious groups, DBSCAN is unable to form fair clusters and is therefore not used in the final application.

D. *PCA & Scatter Plot*

For the initial visualisation of the clusters, PCA is carried out on the tf-idf matrix, which reduces the dimensionality of the matrix from n — where n is the number of feature words — to two, while minimising the loss of information, so that each document can be represented as a point in a two-dimensional scatter plot. The cluster results are then used to visualise the document clustering by plotting the different groups of document points with different colours.

However, the 2D scatter plot is not interactive nor user-friendly. In order to see which point represents which document, the user must use the number annotated with the point on the plot as an index to find the article title in a matrix of article titles printed out in the console. Therefore, a web application is created to visualise the clusters in an intuitive and user-friendly way.

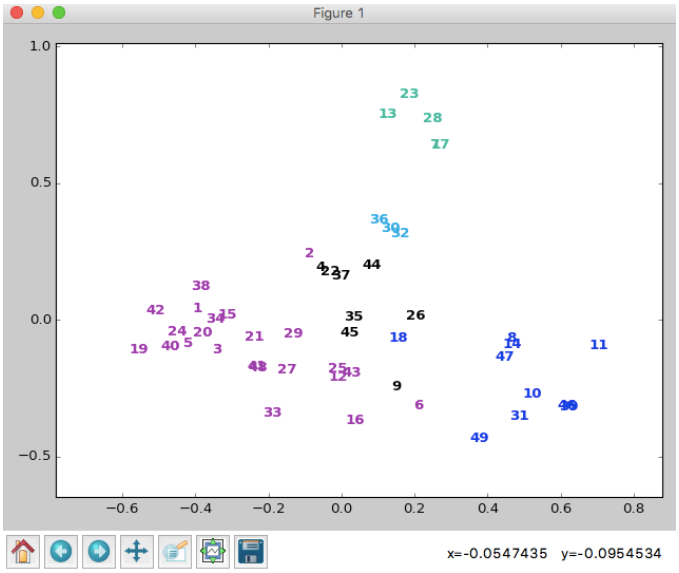

Figure 3 Scatter plot of the document clusters in two dimensions

E. *Web Application*

The web application was developed with the Flask framework. Two routes exist, the index which returns a web page, index.html, rendered from a template; and the clusterer which takes user inputs from the index page and returns a JSON object representing the clustering result to be visualised by D3.js.

Figure 4 The index page with the input form

1) *Route Process*

Index.html contains form elements that take user inputs. It also contains a JavaScript script that attaches a listener to the submit button, so when the button is clicked, the function loadData() would be called. LoadData() displays the loading screen, and calls the function visualise() with the user inputs as arguments. Visualise() subsequently calls the clusterer route function to retrieve the clustering results for visualisation.

2) *Article Retrieval*

When the clusterer route function is called, it gets the user input from the URI and retrieves newspaper articles from the

Guardian. Articles are retrieved using the Guardian API. The function retrieve_articles() in class ArticlesRetriever makes API calls according to the user inputs, which are currently the number articles to retrieve and the date range of the articles. Once the articles are returned in JSON format, each article is turned into an Article object with the title of the article, url, plain body text, and body with html (including images, audio, and video) added on construction. Articles without text, such as Sudoku, are ignored, while live blog articles which contain more than one body content are joined together to form a single article. A list of all Article objects are returned to the clusterer route.

3) Clustering Result Processing

Once document clusters are returned, the function process() in class WebAppProcessor processes the cluster results into a form that can be used by D3.js.

For each article and centroid, a Node object is created with the fields ID, group, features, and body html. The feature words, along with their tf-idf weights for the article, are added to the corresponding nodes in the order from the highest tf-idf to the lowest. For centroids, only the top 10 feature words for the cluster are added. All article objects created are stored in a list to be returned to the clusterer route function.

For each article and its centroid, a Link object is also created to represent the distance of the article from the centroid. Each Link object contains the source node ID, the target node ID, and the distance value as its attributes. The distance measure used is Euclidean distance. The final distance is multiplied by 10, so that the links in the visualisation are long enough to show the difference in similarities between articles in the same cluster. The distance is also incremented by one, so that for clusters with only one article, the centroid node and the article node would not overlap.

All Link objects are added to a list and returned to the clusterer route function.

4) Visualisation

Once the clusterer route function returns the lists of nodes and links in JSON format to visualise(), D3.js is used to visualise the articles, centroids, and their links. Each node is represented by a circle, and each link is represented by a line connecting the corresponding nodes. The distance of each article from their centroid, as well as the thickness of the links both represent how close an article is to the average of the group it is in. When the cursor hovers on a node, the ID of the node is displayed, and when a node is clicked, a side panel is shown on the right-hand side of the window, showing details of the node. For centroids, this includes the centroid ID and the top 10 feature words with their tf-idf weights. And for articles, this includes the title of the article, the feature words with their tf-idf weights, and the article content. Once all components of the visualisation are loaded, the loading screen is hidden and the visualisation is shown.

5) User Interface

The user interface was designed to maximise usability according to Nielsen's 10 usability heuristics for user interface design [31].

Figure 5 Date picker in input form

The form is a natural language user interface [32], which matches a real-world style of asking for information, and acts as self-documentation in that it helps the user easily understand both the purpose of the application and the required inputs.

The input form minimises the user's memory load by making options visible and providing default inputs as examples. When the date field is clicked, a date picker that matches real-world calendars is shown for the user to select the desired dates, making the date appear in a way that is natural to the user and preventing errors.

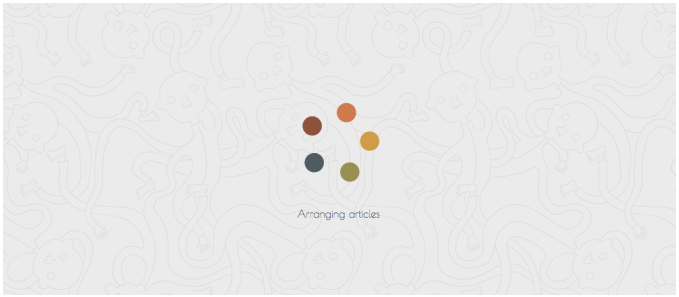

Figure 6 Loading screen

After the user clicks the 'Group!' button, a loading screen is shown, which keeps the user informed about what is going on, achieving visibility of system status.

The article and centroid names are shown in the visualisation when the user hovers above the nodes representing them. This also achieves visibility of system status by giving immediate feedback.

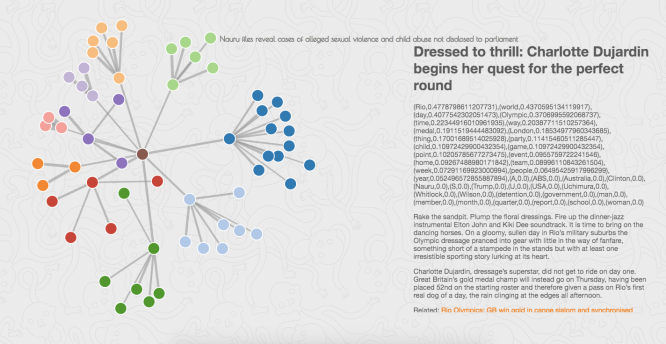

Figure 7 Article title shows on hover

The visualisation allows zoom, and the nodes can be dragged around while still connected through an attraction force between the nodes — matching the inertia in real-world objects.

The user interface is responsive and can be viewed on computer monitors, tablets, and smartphones.

The design is minimalistic while aesthetic, with a subtle paper-like background and slightly muted colours that do not compete with the relevant information.

6) *Deployment*

Heroku is used as the cloud platform for deploying the web application. Gunicorn is used as the HTTP server.

X. Evaluation

Two types of dataset were used to evaluate the results.

The first type is a controlled form of dataset, where obviously similar documents were picked.

The 20 Newsgroups dataset [33] was used initially, which is a collection of messages collected from newsgroups, organised into 20 different topics. The headers of each document are removed, and the documents are renamed according to their categories, which includes comp.graphics, rec.sport.hockey, and alt.atheism. A reduced dataset is used for evaluation, which includes three documents from comp.graphics, three documents from sci.med, and two documents from rec.motorcycles.

Then, newspaper articles from the Guardian were picked and used for evaluation. This dataset includes three articles on the film ’12 Years a Slave’ as well as one article on films in general, three articles on sports, and two articles on technology.

The second type of dataset consists of random newspaper articles within a date range retrieved using the Guardian API. This is the actual dataset used by the final web application, and is much less organised.

A. *20 Newsgroups dataset*

Of the documents, two of the graphics documents, graphics1.txt and graphics3.txt contain an identical section of text. One of the motorcycle documents, motorcycle1.txt is much longer than all of the other documents, and contains special content including an ASCII art and a conversation dialogue, making it distinctive for a human skimming through the documents.

The scatter plot created from the document collection is shown below.

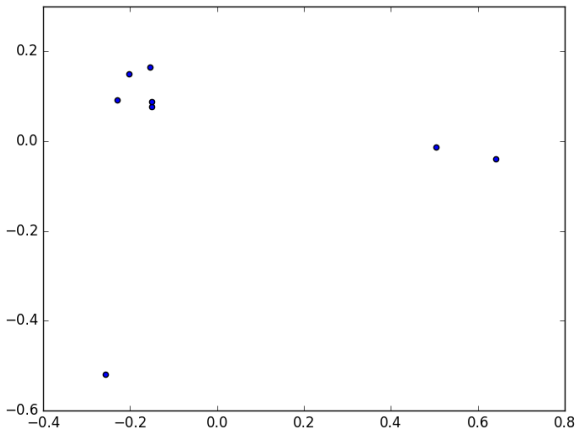

Figure 8 Scatter plot of the documents representation in two dimensions

This scatter plot shows three clusters. The single blue dot represents motorcycle1.txt, and the two blue dots on the far right represent the two graphics documents with a similar section. However, the two dots close to each other in the largest at position -0.14, 0.08 are med1.txt and graphics2.txt, which are quite different semantically.

This showed that the implementation was able to identify the documents which were very similar to each other, as well as documents which were very different. However, the documents which did not contain significant similarities or differences were clustered together.

B. *Handpicked Guardian newspaper articles*

Nine articles from the Guardian picked from three different categories were then used for testing, after the initial pre-processing and tf-idf algorithms were improved. The three articles on ’12 Years a Slave’ as well as the article on films in general were grouped into the same cluster, represented by black in Figure 9. The three sports-related articles were grouped together and represented in purple. And finally the two articles about technology were grouped together, represented by blue.

** Number 10 is not an actual article, but the mean of all articles*

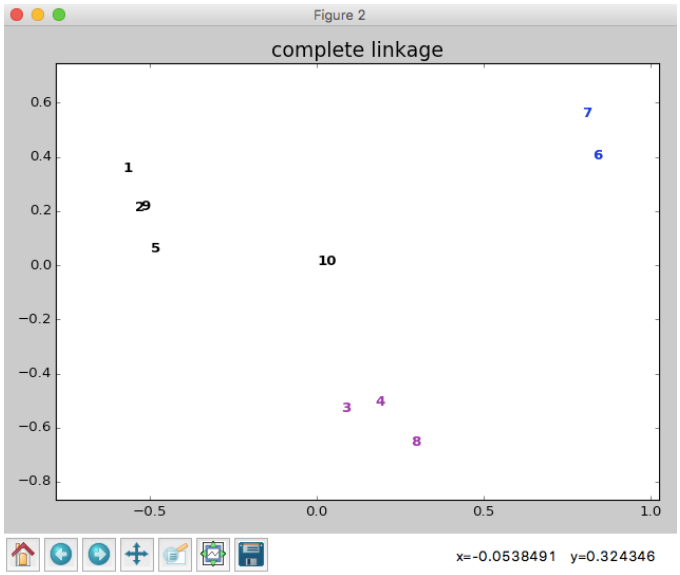

Figure 9 Clustering hand-picked Guardian newspaper articles

This showed that the algorithm was working for articles that can obviously be grouped and distinctively dissimilar.

C. *Dynamically retrieved Guardian newspaper articles*

Finally, real and dynamically retrieved newspaper articles were used to evaluate the clustering algorithm. In one evaluation, fifty articles from 10th July 2016 to 10th August 2016 were retrieved and grouped into 10 clusters. The following groups were formed:

3 articles related to finance and economy
13 articles related to the Olympics
6 articles related to politics, census, and government
3 articles on Theresa May
2 articles on prisons and 2 pop-culture reviews
1 article on celebrity, and 1 article on Twitter
5 articles on police
5 articles on sports
4 articles on Nauru, 2 articles on medicine, 1 article on Muslims
1 article on corrections to past articles

This shows that most of the articles that are semantically similar were grouped together, with a few outliers incorrectly grouped.

XI. Future work

There are many opportunities for future work to extend this application, many of which were not implemented due to time constraints. Some of these include:

- Adding semantic meaning of the words using Word2Vec
- Research and implementation of different methods for finding the correct number of clusters, such as Elbow method and Information Criterion Approach
- Research and implementation of different clustering algorithms, such as X-means [34] and Lingo
- Research and implementation of supervised clustering algorithms
- Implementation of additional visualisation types

Further validation and testing could also improve the accuracy and speed of the current clustering algorithm.

XII. Conclusion

The project goal was to implement an application which, given a data set of newspaper articles in text format, groups the documents into clusters based on their semantic meaning, and visualises the clusters.

Evaluation shows that the web application is able to group articles covering similar topics into the same clusters, and that the application is able to visualise the clusters in an intuitive and user-friendly way. This highlights the potential for Clusterr to become an effective application for news readers to quickly identify and read newspaper articles that cover similar topics.

XIII. References

[1] Northern Light, LLC, “The basics of Northern Light market intelligence solutions,” Northern Light, LLC, 10 September 2016. [Online]. Available: https://northernlight.com/market-intelligence-solutions/. [Accessed 10 September 2016].

[2] S. Ilya, “Yippy,” Bourabai Research, [Online]. Available: http://bourabai.kz/dbt/yippy/english.htm. [Accessed 10 September 2016].

[3] Noggle, “NoggleMap Search Document Clustering,” Noggle, [Online]. Available: https://www.noggle.online/knowledge-base/document-clustering/. [Accessed 10 September 2016].

[4] Carrot2, “Carrot2 Clustering Engine,” Carrot2, [Online]. Available: http://search.carrot2.org/stable/search. [Accessed 10 September 2016].

[5] Carrot2, “Algorithms,” Carrot2, [Online]. Available: http://project.carrot2.org/algorithms.html. [Accessed 10 September 2016].

[6] NLTK Project, “NLTK 3.0 documentation,” NLTK Project, 9 April 2016. [Online]. Available: http://www.nltk.org/. [Accessed 9 May 2016].

[7] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa and A. Mueller, “API design for machine learning software: experiences from the scikit-learn project,” *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases,* pp. 108-122, 1 September 2013.

[8] Numpy developers, “NumPy,” 2016. [Online]. Available: http://www.numpy.org/. [Accessed 9 May 2016].

[9] A. Ronacher, “Flask,” Pocoo, 2016. [Online]. Available: http://flask.pocoo.org/. [Accessed 12 September 2016].

[10] Gunicorn, “Gunicorn,” Gunicorn, [Online]. Available: http://gunicorn.org/. [Accessed 12 September 2016].

[11] M. Bostock, “Data-Driven Documents,” D3js, 2015. [Online]. Available: https://d3js.org/. [Accessed 12 September 2016].

[12] The jQuery Foundation, “jQuery,” The jQuery Foundation, 2016. [Online]. Available: https://jquery.com/. [Accessed 12 September 2016].

[13] Timofey, “AIR DATEPICKER,” Github, [Online]. Available: http://t1m0n.name/air-datepicker/docs/. [Accessed 12 September 2016].

[14] M. D. Christopher, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2008.

[15] The Stanford Natural Language Processing Group , “Stanford Named Entity Recognizer (NER),” Stanford NLP Group, [Online]. Available: http://nlp.stanford.edu/software/CRF-NER.shtml. [Accessed 10 September 2016].

[16] E. T. K. Sang and F. D. Meulder, “Language-Independent Named Entity Recognition (II),” CLiPS Research Center, 5 December 2005. [Online]. Available: http://www.cnts.ua.ac.be/conll2003/ner/. [Accessed 10 September 2016].

[17] S. Polamuri, “Five most popular similarity measures implementation in python,” 11 April 2015. [Online]. Available: https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/. [Accessed 9 May 2016].

[18] A. Huang, *Similarity Measures for Text Document Clustering,* Hamilton: Department of Computer Science; The University of Waikato, 2008.

[19] P.-N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, Boston: Addison-Wesley Longman Publishing Co., 2005.

[20] M. Matteo, “A Tutorial on Clustering Algorithms,” [Online]. Available: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html. [Accessed 9 May 2016].

[21] D. Sculley, “Web-Scale K-Means Clustering,” in *WWW '10*, New York, 2010.

[22] J. Béjar, “K-means vs Mini Batch K-means: A comparison,” 17 May 2013. [Online]. Available: http://upcommons.upc.edu/bitstream/handle/2117/23414/R13-8.pdf. [Accessed 12 September 2016].

[23] Frontline Solvers, “HIERARCHICAL CLUSTERING,” Frontline Solvers, [Online]. Available: http://www.solver.com/xlminer/help/hierarchical-clustering-intro. [Accessed 9 May 2016].

[24] J. H. Ward, “Hierarchical Grouping to Optimize an Objective Function,” *Journal of the American Statistical Association,* vol. 58, no. 301, pp. 236-244, 1963.

[25] The Pennsylvania State University, “14.4 - Agglomerative Hierarchical Clustering,” The Pennsylvania State University, [Online]. Available: https://onlinecourses.science.psu.edu/stat505/node/143. [Accessed 12 September 2016].

[26] J. Sander, M. Ester, H.-P. Kriegel and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Data Mining and Knowledge Discovery,* vol. 2, no. 2, pp. 169 - 194 , 1998.

[27] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics,* vol. 20, pp. 53-65, November 1987.

[28] L. I. Smith, “A tutorial on Principal Components Analysis,” 26 February 2002. [Online]. Available: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. [Accessed 12 September 2016].

[29] M. Liberman, “Penn Treebank P.O.S. Tags,” 2003. [Online]. Available: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html. [Accessed 12 September 2016].

[30] Scikit-learn, “Clustering,” Scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/clustering.html#different-linkage-type-ward-complete-and-average-linkage. [Accessed 12 September 2016].

[31] J. NIELSEN, “10 Usability Heuristics for User Interface Design,” Nielsen Norman Group, 1 January 1995. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 12 September 2016].

[32] B. Long, “Natural Language as an Interface Style,” University of Toronto, May 1994. [Online]. Available: http://www.dgp.toronto.edu/people/byron/papers/nli.html. [Accessed 12 September 2016].

[33] J. Rennie, “20 Newsgroups,” N/A, 14 January 2014. [Online]. Available: http://qwone.com/~jason/20Newsgroups/. [Accessed 9 May 2016].

[34] D. Pelleg and A. W. Moore, “X-means: Extending K-means with Efficient Estimation of the Number of Clusters,” in *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, 2000.

[35] J. B. MacQueen, “Some Methods for classification and Analysis of Multivariate Observations,” in *5th Berkeley Symposium on Mathematical Statistics and Probability*, California, 1967.

[36] N. Kaur, “A Tweet Grouping Methodology Utilizing Inter and Intra Cosine Similarity,” in *IEEE 28th Canadian Conference on Electrical and Computer Engineering*, Halifax, Canada, 2015.