ALGORITHMIC ‘IDEA’ CLUSTERING VIA NATURAL LANGUAGE PROCESSING

Niall Curtis – C1623580

Supervisor: Alun Preece

1 Abstract

Crowd-sourced innovation at scale produces a large number of ideas, which must be traversed to transfer the knowledge to someone that can act upon the ideas. The length of time required to view all the ideas, uncover the trends among the data, and sift through duplicate or “bad” ideas often makes the process formidable or inaccessible to some establishments.

This project aims to create a system that uses natural language processing and exploratory data mining to automate some of this process – uncovering trends automatically, grouping duplicates and extracting key content from ideas.

The project will also reduce the visual burden of a long list of ideas by using the analytic data to create an attractive visualisation that engages the user and captivates them to spend time understanding the ideas.

2 Table of Contents

3 Table of Figures

4 Introduction

4.1 The Aim

During my year in industry in second year and since then, I worked for Simply Do Ideas[1], a company that builds web software to handle crowd-sourced innovation. Their application facilitated innovation through a challenge process – a stakeholder would propose a question; for example, "How can we improve the office?" and employees could submit their ideas using a proposal template.

The challenge/idea model is propped up by social engagement, where other employees can like and comment on each other's ideas to push forward those that they feel will be the most impactful to them. The side effect of this process was that challenges could amass a large total of ideas. In some cases, larger organisations could have hundreds of ideas, many of which uncompleted, duplicated or misunderstood the question. The "challenge owners" must then devote a significant amount of time to sifting through the pile of ideas to find actionable information.

This is a human bottleneck in a product that's main aim is to decentralise the innovation product and move it to a digital environment. The goal of this project is to use modern text analysis tools to streamline the post-ideation process and decrease the time required to analyse a set of ideas from a challenge. This will be achieved by highlighting trends in the dataset, grouping ideas on similarity, assessing duplicates and reducing the content that the challenge owner must ingest. This data will then be presented in an easily digestible, visually stimulating manner, as it is vital for the data to be actionable by the user.

4.2 Intended Audience

This project is done in tandem with Simply Do Ideas, with the idea that the analysis tool will be usable and accessible by anyone that wishes to run a challenge-based ideation project. A number of the staff members within the company have cooperated with organisations to run innovation challenges, so this project will be built to alleviate the stress points highlighted by these staff members.

My project can be qualitatively evaluated against their difficulties analysing a set of ideas and how the technology solves those difficulties, along with the trends and results they picked out manually and will highlight the difference between automatic and manual analysis – both positively and negatively, showing perhaps where technology fails to understand ambiguity, intention, sentiment and personal feeling.

4.3 Scope

The scope of the project for its current duration is to develop a system that operates independently of the company's existing product ecosystem. The basic scope to prove the 'proof-of-concept' of the proposal is a backend analysis system that will accept an input dataset of ideas, perform the proposed text analysis and output a revised dataset with the data from the analysis added to the ideas.

[1] https://www.simplydo.co.uk/

As a counterpart to this, there will be a web-based user interface that enables the end-user to upload their set of ideas, which will use the analysis data to visualise the results in a stimulating manner that easily enables the user to understand the analysis and fully demonstrates the capability of the backend.

These two elements combined provide a strong minimum value product to demonstrate the proposal of the project, providing a good basis on which to evaluate its success and compare different methods of analysis.

4.4 Outcomes

A successful outcome of this project would be that, with some consensus, the idea analysis tool is qualitatively evaluated to reduce the man hours and improve the quality of the post-ideation process – crucially, reducing the human bottleneck in a digital process, and increasing the value of the overall product to the consumer.

5 Background

5.1 Wider Problem Context

In order to properly understand the problem in question, it is important to visualise the post-ideation process. The following diagram is what, typically, a challenge owner will see following the completion of a challenge process, when there is a number of ideas submitted. (A challenge owner in the context of this report is an individual of group of stakeholders that proposed the challenge, and would manage the implementation of ideas on completion).

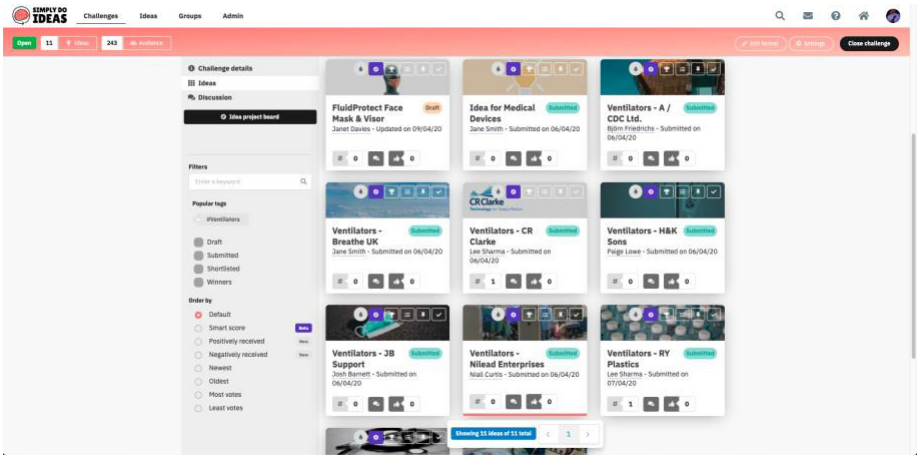

Figure 1 - Set of ideas in a challenge in Simply Do Ideas

This challenge is taken from an internal company wide one – A larger organisation may have dozens of pages with hundreds of ideas.

The web application has alleviated the logistically difficulty of manually collecting employee suggestions, previously done via suggestions boxes or emails. However, sorting through these ideas is still difficult.

Simply Do attempts to alleviate some of these difficulties by providing tools to sort ideas, filter by keywords, mark them with filters to indicate they've been seen, and many more. The stress point here is still that every single idea must be manually clicked into, read, and actioned in some way (marked as read, tagged, commented on, some way to imply that it has been digested).

This issue is further highlighted in large datasets, where a portion of ideas will be partial duplicates of each other, incomplete, or have misunderstood the question in some way.

Taking the time to inspect a "bad" idea can be considered wasted time, which adds up over a large dataset. The other issue encountered is that when challenges are evaluated by a group of people, trends in data tend to be missed. This is especially noticeable when a set of ideas is "rationed" in groups for people to look into.

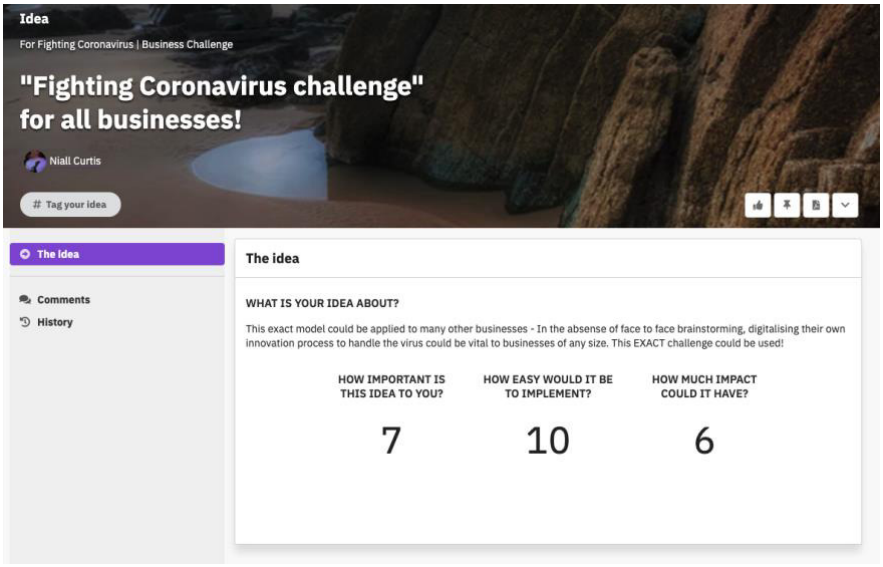

Figure 2 - Example of an idea in the web application

These two areas are where the web application currently lacks “intelligence” in the sense that they are still largely manual processes. These issues were specifically highlighted by two organisations that run a number of challenges with a large (multiple thousands) userbase – Development Bank of Wales and South Wales Police. Providing tools to improve this process is what the project will attempt to achieve.

5.2 Stakeholders

The project will be completed in tandem with Simply Do Ideas and will be involved in regular communication to inform progress and iteratively review progress.

The problem for the project was identified as a result of customer surveys with a number of the company’s clients, including those mentioned in the previous section. As the company offers direct challenge support with a business’ ideation department as part of a license fee to use the application, employees that have collected this feedback will be used as representative stakeholders for the problem. Customers’ employees will therefore be the basis for qualitative evaluation of success.

The company also has internal challenge analysis reports, where the supporting employee sifts through the ideas in tandem with the challenge owner manually and extracts duplicates, trends and keywords. These reports will be compared with the output of the project.

5.3 Theory

If the solution is to ultimately reduce the time taken to understand a set of ideas, then its methodology for visualisation should immediately be familiar. Therefore, the product should be planned top-down – where the best method of presentation is decided upon, then the analysis system is built to support it.

Broadly speaking, the challenge and idea process produces ideas that are unseen – They are unfamiliar to challenge owner, and in the case of some analysis tool, unfamiliar to the machine or algorithm. Analysis can be disseminated into two categories; confirmatory, which is the idea of confirming a previous hypothesis, and exploratory, which is discovering characteristics of a data sets and involves summarisation and visualisation. (Heck 1998). As

these challenge analysis activities involve unseen datasets, the author will investigate exploratory methods of analysis.

Research suggests that our brains are naturally prone to categorising things (Branan 2010); This supports the decision to make any visualisation and presentation as "natural" as possible, to imitate and streamline what a user may consider the "natural" way they would manually filter through a dataset.

An internal company activity that looked at how someone may solve the problem of sorting through a large dataset concluded that most users begin by applying some degree of categorisation to reduce the problem into smaller tasks. A standard method of grouping comes in the form of cluster analysis as the theoretical method of analysis. Another popular method is word embeddings to generate vectors from words.

5.3.1 Cluster modelling

Centroid-based clustering is where isolated groups are represented by a central point that may not be part of the data, but can be used to display representative characteristics, which allow group summarisation.

For this project, it is appropriate to partition ideas by their content. See Figure 3 for a visual example of partitioned observations using a k-means cluster model (Chire 2011).

In general terms, k-means centroid cluster modelling aims to partition items such that they belong the cluster with the nearest centroid. In this model, k is a pre-determined number of distinct clusters to partition observations into (MacQueen 1967).

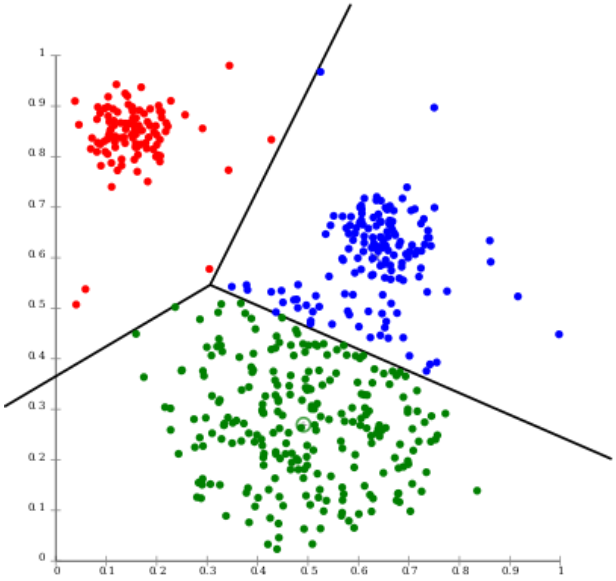

Figure 3 - K-Means Gaussian Data, Chire, 2011.

This task can be applied to text documents by using common algorithms to represent sentences numerically, such as TFIDF (Term Frequency – Inverse Document Frequency). Term Frequency is the ratio of the current word to the number of all words in the document, and Inverse Document Frequency measures the ratio of the documents in the collection that contain the given word. TF-IDF is then the product of these two values to uncover the "important" of the term (Ramos 2003). The TF-IDF of a document can then be represented in a two-dimensional space and partitioned into clusters.

Reducing cognitive load is a key task for the project, however using text processing also makes rich data analysis accessible. Trends can be deduced automatically; a challenge owner can understand the key themes from the idea set before they begin manually inspecting ideas. Practical implementations of this can be very basic, such as simple word frequency, but richer analysis can be done with text-mining techniques, such as the statistical topic model.

5.3.2 Topic model via LDA

Topic modelling is an unsupervised, exploratory machine learning technique that attempts to uncover trends (topics) that can characterise a document or set of documents (Rehurek and Sojka 2010), through numerical representation. Trends are calculated by the occurrence of terms in a document. A popular approach to obtain topics is LDA (Latent Dirichlet Allocation); an algorithm that operates on an assumption that documents have a mixture of topics, and words are generated based on a probability distribution.

5.3.3 Doc2Vec

Another implementation for numerical representation for words. An extension of the word2vec model (Goldberg and Levy 2014), which is a numerical representation of words that is able to retain the "meaning" of words when considering their closeness, including similarities such as analogies, synonyms etc.

For example, the relative relationship from king to man is gender, whereas king to queen is monarchy. Given woman as well, word2vec can reason that king – man + woman = queen, through its understanding of the meaning and relationship between words. The doc2vec model (Lau and Baldwin 2016), extends this by using document vectors to represent the concept of sentences, rather than just words. It uses the understanding that the order of words in a sentence affects the meaning of the sentence when looking at relationships.

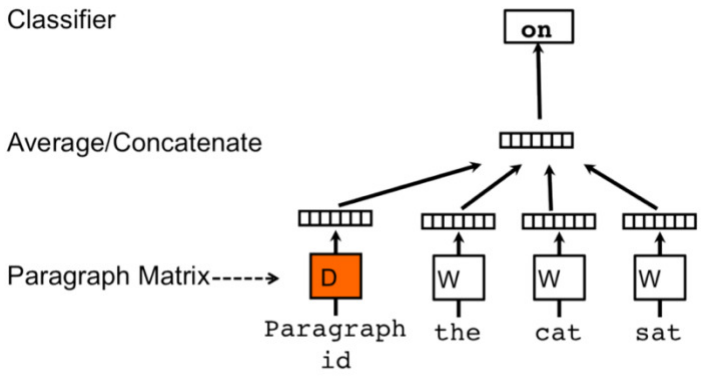

Figure 4 - Doc2Vec Vectors, (Budhiraja 2018)

5.4 Existing solutions

Internal competitor research within Simply Do Ideas has highlighted the approaches that a number of competitors in the ideation-innovation space in the United Kingdom have taken to streamline the post-ideation process for challenge owners.

The research looks at the data reporting for five competitors, and what methods they use for idea analysis. Of the five companies, four of them use self-assessment to group and sort ideas. This generally involves users attaching their own identifiable metrics to ideas with a rigid structure. For example, this may be their estimation of simplicity to implement an idea, the cost

it might save on implementation, or a set of tags. This pushes the burden of categorisation onto the user, still remaining largely a manual process but reducing the requirement for the challenge-owner to do all the work. Three of the companies offer an implementation of NLP (natural language processing) for their reporting. One of them states that they provide AI to "analyse unstructured data ... to consolidate similar ideas and connect people interested in like-minded ideas", which suggests some similar clustering approach. One other states they use NLP for identifying duplication, and the final uses sentiment analysis.

Of the five competitors, none are understood to use any form of two-dimensional representation of idea content, direct centroid-based clustering or topic model analysis. The author hopes that using this form of representation and analysis would, if taken further, provide Simply Do Ideas with a competitive advantage in this area where no competitor has ventured into, at the time-of-writing.

Clustering text from digital sources, in itself, is well explored, and has myriad implementations. Common exploratory analysis can be done where masses of data are available freely in some digital space such as Twitter, where the author can source a potential dataset of millions of sentences.

A typical use case may be attempting to group politically focused tweets to find large unseen trends and communities, which is particularly applicable due to inherent tribalism in this topic. (Koç et al. 2018). While used extensively elsewhere, the discussion of clustering in the ideation space, at least as far as scholarly and competitor research goes, is seemingly unexplored.

While the techniques used in this project remain largely equivalent to usual cluster modelling, there is no clear precedent for visualisation, which this project will hinge much of its success on. Additionally, the project will look at how a combination of the aforementioned techniques can improve the richness of analysis.

5.5 Possible constraints

Users of the Simply Do platform tend towards being less experienced with computers and managing large datasets, so the possible complexity of detailed analysis can in fact increase the cognitive load of managing a challenge. This would have the inverse effect of increasing the time requirement for challenge owners. It is vital therefore that the project focuses on a straightforward operation, that uses detailed analysis to reduce, rather than increase, the complexity of the process. A successful outcome would not be simply successful data analysis, but rather successful packaging and conveying of data analysis.

The quality of data also provides a hurdle in natural language processing. Ideas have no baseline for correct grammar, spelling or sentiment. The opposite is often preferred, for a faster user experience – quickly submitting short ideas. By promoting this type of ideation, sacrifices are made on the calibre of the text itself. The content will also typically discuss the challenge set forth, and as such many ideas may have similar content if they begin sentences with something akin to "I think we can improve the office by…"; leading office to be a reoccurring theme across all ideas. Calibration of techniques will therefore be important to avoid ideas being clustered or grouped on central themes that are unhelpful to the challenge owner. This also extends to usage of the words "idea", "challenge" and other terms specific to this subject area. There must be a thorough list of terms to ignore, as the efficiency of the k-

means algorithm increases as clusters are more isolated, given cluster isolation p, with a run time of O(kn/p2). (Kanungo et al. 2000)

The size of the average idea set is also not ideal for detailed processing. A typical corpus for text analysis would contain thousands of entries, whereas this system will manage tens to hundreds. The accuracy of results and isolation of clusters scales with the number of items to analyse, but other research suggests that there is no explicit "minimum" size for cluster analysis (Siddiqui 2013), which suggests that the chosen method will still be applicable to this use case.

5.6 Research Questions

This project will consider the viability, usability, and achievability of using NLP and exploratory data mining techniques to streamline and enrich the process of navigating and managing a large number of ideas. The project will investigate and justify appropriate methods for both analysing and visualising data and evaluate the extent to which the software produced successfully improves this process.

The key metrics will be demonstrating that the stakeholders previously identified:

a. find the data activities performed on the idea set noticeably reduces the cognitive load navigating the idea set and
b. reduces the time needed to draw conclusions, and identify relevant themes.

6 Specification and Design

6.1 Approach

The project has two distinct elements – The "backend" system, which handles the analysis, and the "frontend" system, which handles visualisation. The backend will take a set of ideas, perform the data mining activities to uncover the unseen characteristics of the set, and return the set of ideas with the new knowledge included.

The frontend will be able to reason the output from the backend and use appropriately justified methods of visualisations to make the knowledge easily interpretable by the end user. The two systems will operate as black boxes – while they will function together as a system, they will not be explicitly intertwined and able to operate independently, similar to a client-server model.

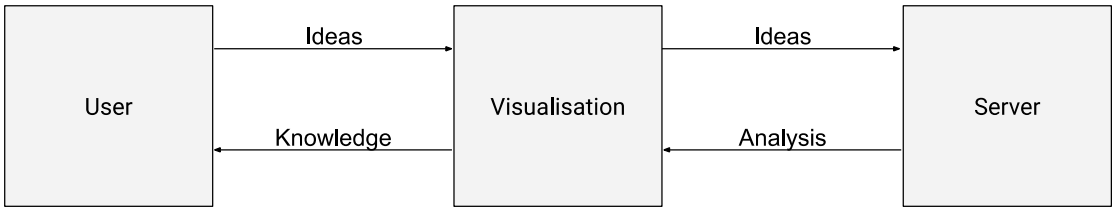

Figure 5 - Transfer of information in the project

6.2 Development Process

This black box system is in aid of an Agile, iterative development process with the ability to independently work on the analysis and visualisation tools. Upon regular liaison with stakeholders from Simply Do Ideas, improvements can be made in "sprint" intervals, and focus shifted where necessary. The goal with having two independent systems as part of the project facilitates having distinct development processes for each constituent.

Reviews of the entire project have been planned with the Simply Do Ideas team as a whole, and evaluations with SME (Subject-matter Experts) about how the project could have aided challenges they have run in the past.

Before beginning the development process, the author researched a number of technical implementations of the techniques discussed, as well as previous applications of the technologies to learn the processes involved. Great attention was paid to the scikit-learn Python package (Pedregosa et al. 2011). This provides easy-to-use APIs for a number of popular machine learning algorithms and is especially relevant to this project as it includes implementations for k-means clustering, TF-IDF, and LDA topic modelling.

Additionally, the author devoted time to a popular Python handbook called "Python Data Science Handbook" (VanderPlas 2016) that teaches usage of these implementations and how to maximise their potential. The book covers, step-by-step, methods for data mining and visualisation.

6.3 Requirements

Requirements have been formulated by the previously mentioned Simply Do Ideas team members acting as stakeholders on the part of license-holding clients. From the project aims, I devised and verified a set of explicit milestones that would shape the application and

provide a basis for evaluation. The milestones were then split into minimum and desirable, in order to prioritise development progress effectively, and across three separate categories – Supplementary tasks, Analysis, and Visualisation. These milestones were included in the initial report and will be reiterated here to aid review.

Requirement	Acceptance Criteria
Key Supplementary Tasks	
Data Clean-Up (MINIMUM): System must be able to accept and normalise a large variety of ideas in different formats, to build a standardised corpus of ideas for further comparison, amalgamating the content for text processing.	System successfully converts an unsorted set of ideas to an array of documents with identical keys, and correspondingly formatted items.
Analysis	
Supports Multiple Use Cases (MINIMUM): The system will support a variety of different skill-levels, supporting a multitude of use-cases. The system will support the user regardless of their experience.	System has a set of features with varying degrees of involvement and complexity. At a minimum, the system will have two workflows, one that is heavily automated and one that allows the user to manually explore the rich analysis.
Word Frequency (MINIMUM): Word and bigram frequency analysis; allowing simple visualisation of the most used terms. Both segmentation and tokenization will be used, along with a common and user-extended library of stop-words to remove analytically useless connectives, verbs and common terms that have no use for investigation	System outputs an array of the most popular terms from the set of ideas, along with how many times they have been used, enabling creation of an appropriate visualisation technique such as a word cloud.
TF-IDF (MINIMUM): Analytic techniques to pick out the most 'popular' terms in the set of ideas, which in turn allows discovery of the overall 'key topics'. Acts as basis for clustering and enables broad stroke inspection of challenge outcomes.	System produces a vectorised TF-IDF matrix for the ideas in the dataset, and reduces them to a two-dimensional representation, giving all ideas both a TF-IDF value and TF-IDF "coordinates" to be plotted.
Clustering (MINIMUM): Act of separating the ideas into previously unknown, potentially non-discrete groups based on their individual TF-IDF matrices, traits, and the occurrent of trends within the ideas; the aim being to create subsets of ideas that belong to a particular topic within the idea corpus, which were previously found via inverse document frequency analysis. This is the end goal to solve the post ideation process of filtering ideas, as the administrators can immediately extract and dissect the key outcomes of the challenge.	System uses natural language, exploratory cluster analysis to partition all ideas into discrete groups, using an appropriately justified technique, such as k-means. Each idea is given a cluster based on the predetermined number of clusters, and this can be visualised along with the TF-IDF plots. Clusters are verified in their correctness via qualitative evaluation.

Word Embeddings and Idea Closeness (DESIRABLE): Representing individual words as vectors in a vector space, finding words that have the same representation and thus being able to define some degree of similarity, or closeness in the vector space, between different ideas. Using a pre-existing populated embeddings model for word comparison, we can compare the features of ideas and be able to produce a traversable map of ideas.	The content of ideas is converted by the system into a numerical representation using an appropriately justified method, such as doc2vec. The model will allow suggestions of similar ideas, and discovery of the closeness of given ideas to others. This can also be verified in effectiveness through qualitative evaluation.
Visualise	
Key topics (MINIMUM): Simple visualisation of the key topics acquired from TF-IDF and term frequency analysis data, to enable a minimum viable product	TF-IDF output from the analysis system is visualised appropriately, with justification and verification from evaluation. The applicability of conversion from TF-IDF matrix to two dimensional coordinates makes scatter plot a suitable candidate to visualise the TF-IDF differences between ideas; but any way to visualise the TF-IDF scores would be a measure of success.
Clustering representation (MINIMUM): A well-justified method to represent the clustering of ideas in the dataspace, that fully supports a fast and efficient idea sifting process for the user. Clearly define the extracted key topics and allow granular access to ideas clustered within these topics for deeper analysis.	The visualisation successfully differentiates between ideas of each cluster. This visualisation paradigm is evaluated to prove that the differentiation is easily identifiable. Within the discrete clusters, the visualisation will provide controls to further limit the number of visible ideas.
Embedding graph (DESIRABLE): Take advantage of the embedding vector-space and similarity data to create a linked map visualisation of the closeness of ideas. Can pick out trends between clusters, navigate the data space, view potential outliers and more.	Closeness of ideas have a suitable exploratory visualisation method. The chosen vector space model has a module to show which ideas are the closest, in a way that it "suggests" the next move of the user.
Data manipulation (DESIRABLE): Create set of tools that give the user the ability to manipulate the tools that analyse the dataset in real time to allow them to decide their own granularity of analysis – By empowering them to complete actions such as adjusting cluster size, with the visualisation robust enough to support these user activities.	The user interface has controls that interact with the analysis system as well as being able to visualise the result. Controls are verified to improve the richness of analysis, and are simplistic enough as to not impede the progress of the user. Any granularity is optional, allowing power users to adopt if wanted.

6.4 Requirement Acceptance Criteria

When evaluating the success of a project, two potential schools of evaluation are qualitative and quantitative (Newman et al. 1998). Generally, quantitative research is applicable when confirming a theory or assumption, expressed through numbers; and can be used to create facts. Qualitative evaluation is expressive, naturally opinionated and enables exploration and generating insight. It is possible in many cases to combine both qualitative and quantitative to produce a more thorough picture.

The value of this project is in the perceived improvement of the post-ideation experience for the challenge administrator. Whilst the project aims to reduce the time spent understanding a set of ideas, the provision of additional analysis tools means this criterion is more complicated, as one may spend more time uncovering hidden trends that adds value to their reporting.

The system in itself produces results through exploratory data mining exercises – k-means, topic modelling – which aim to generate previously unseen information as opposed to confirming a pre-existing hypothesis. It fits into a pre-existing application that's business model is exploratory innovation via ideation. As such, in the majority of cases, there is no hypothesis to evaluate the effectiveness of project's analysis in a confirmatory manner. This calls for the need for exploratory evaluation.

The requirements can be appraised via a qualitative evaluation with SMEs (Subject Matter Experts) from Simply Do. Chosen SMEs can be asked to choose a previously completed challenge that they have overseen in the past, where they have manually analysed and reported on trends and information from the ideas, where this data and their personal experience of the process will be qualitatively compared to conducting a similar process with the aid of this project. Additionally, general evaluation can be performed with a wider selection of team members to understand opinion on the user interface and basic effectivity of the analysis.

A possible solution for quantitative assessment of the requirements could be a confirmatory experiment with an unseen set of ideas. The time taken to reach an agreed "solution" (a suitable report of the outcome of the ideation process) can be compared between an unaided manual report, and a report created with the aid of the project. Unfortunately, unforeseen circumstances have rendered this method of evaluation infeasible. Office closures due to SARS-CoV-2 (Organization 2020) stretched company resources thin and made in-person contact unattainable, reducing the possible scope of evaluation with subject matter experts. For the duration of this project, all communication can only be done digitally, with less evaluation contact time than hoped upon inception of the project.

6.5 Technologies Used

This section will overview the most influential technologies and programming languages used. Only the most important modules will be included that are necessary for the project to function, with tertiary components excluded to reduce complexity e.g. Built in functionality such as numpy for Python will be excluded here, and mentioned where necessary in the Implementation section.

- **Python:** Used as the basis for the analysis system in a client-server model. Chosen for its simplicity and readability in the execution of calculation, and widespread support for common data mining applications.
    - **Flask:** Python web application framework (Grinberg 2018), designed for simplicity in getting started while remaining extensible.[2]
    - **Scikit-learn:** Widely used, free machine learning library for Python. Contains implementations of, and simplistic APIs for a variety of data mining models, including k-means clustering, LDA topic model and TF-IDF.[3]
    - **Gensim:** Python library that specifies in unsupervised topic modelling and natural language processing. Has capability to support word2vec and doc2vec.[4]
- **React:** Component-based JavaScript library for building web user interfaces. Supported through a wealth of community extensions.[5]
    - **Semantic UI**: React implementation for the vocabulary-based Semantic UI visualisation library. Uses straightforward component model to reduce time spent designing basic UI components such as Buttons.[6]
    - **NIVO:** Deep data visualisation components for plotting information in React, build on top of the barebones d3.js JavaScript data visualisation library.[7]

6.6 Static Architecture

Broadly, the static architecture is divided into four distinct steps, as the ideas are reduced into a more manageable format.

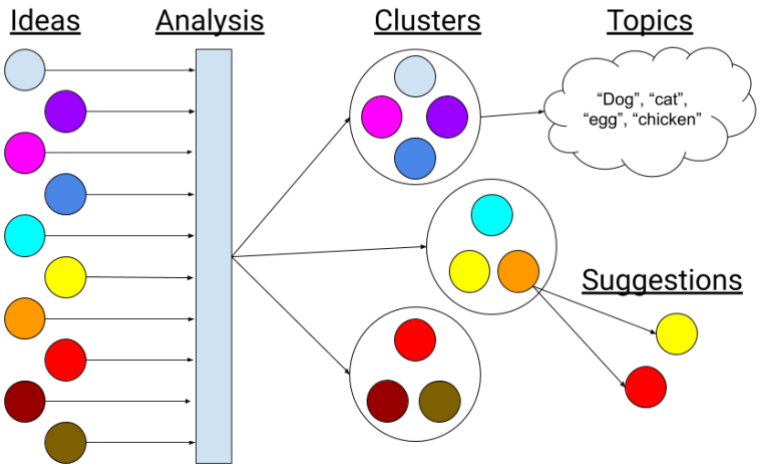

Figure 6 – Idea analysis flow diagram

The broad-scope architecture flows from left to right in Figure 6, starting with the unsorted corpus of ideas that begins following the completion of a challenge in Simply Do Ideas, once a collection of ideas has been produced. The set of ideas is sent to the analysis system, where the data mining exercises – clustering, TF-IDF, topic model etc – are performed. The system then outputs the newly analysed corpus of ideas with the cluster information in place. In the

2 https://palletsprojects.com/p/flask/

3 https://scikit-learn.org/stable/index.html
4 https://radimrehurek.com/gensim/
5 https://reactjs.org/
6 https://react.semantic-ui.com/
7 https://nivo.rocks/

visualisation, the set of ten unorganised ideas are initially reduced to three discrete clusters, the first step in reducing the cognitive load. From there, the clusters are further simplified through two parallel processes; extracting unseen topics from the clusters to identify key themes, and using vector space closeness to suggest similar ideas from a singular one. This simplistic overview can be elaborated upon by considering the static architecture from two viewpoints; the analysis and visualisation. The overall ethos and fundamental loop of the system is:

1. **Analyse:** Cluster, topic model, find popular terms
2. **Uncover**: Discover hidden trends among data, traverse the "space" of ideas, find similar suggestions, understand what is popular
3. **Delve:** Using the knowledge uncovered, increase the granularity of the analysis to repeat the process and expose further information

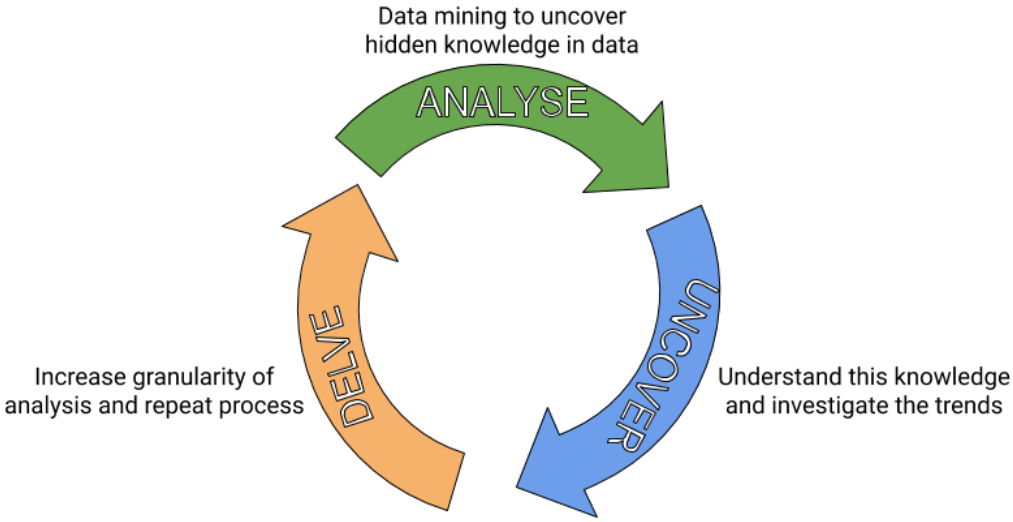

Figure 7 - Analyse Uncover Delve Loop

These three crude stages can be repeated until the desired level of granularity is achieved. The project aims to provide an effective solution to the problem within one loop, with enough data to draw solid conclusions – while providing the option to continue the cycle when desired.

6.7 Static Architecture (Analysis)

The following flow diagram, Figure 8, demonstrates the route data will take from the conclusion of a challenge to the analysed dataset used for the visualisation.

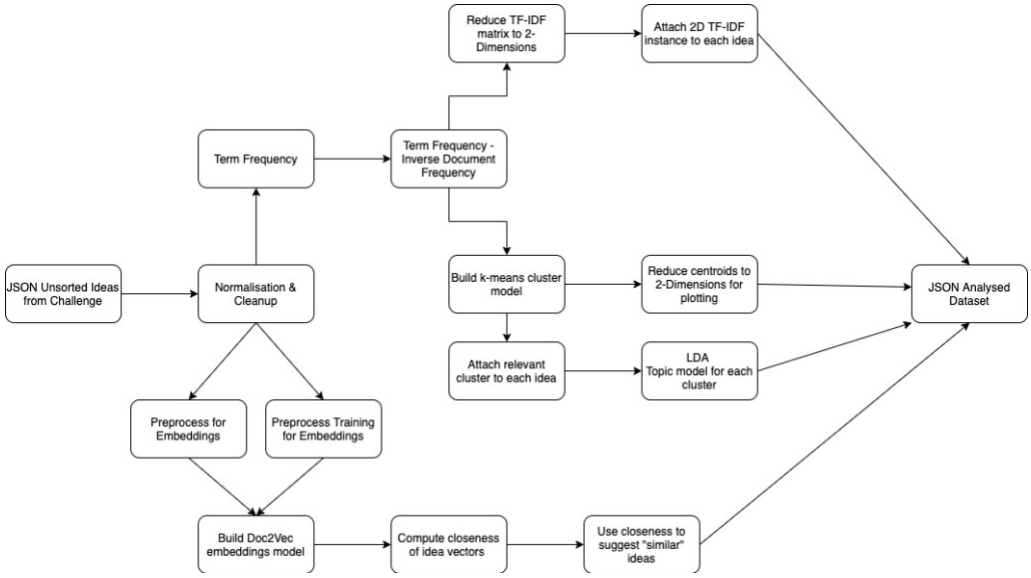

Figure 8 - Analysis Data Flow

Figure 8 begins with an unsorted JSON format file, containing the ideas from any given challenge. This is the standard output format used in the Simply Do Ideas application, due to the data being stored in a document format using MongoDB[8]. Upon receiving this data, the first step for the system is to clean-up and normalise the ideas, to enable the data mining activities through a consistent data structure with additional metadata. This process will also involve the removal of stop words. Stop words are analytically useless words, that are equally likely to be in irrelevant documents as they are to be in relevant documents when doing comparisons. This can include connectives and articles such as “and” or “the”. The normalised data will have the following fields:

- _id: Unique identifier for each idea, which can be used to refer back to the original unsorted corpus
- Name: Name given to an idea by a user
- Content: Single string amalgamation of all the content fields from an idea, which may be spread across nested objects in the original document
- Tokens: Array of strings contained each word in the Content section, with stop words removed
- Stems: Stemmed copy of the Tokens array. A stem is a word without any suffixes/prefixes to reduce it to its root. E.g. running to run.

More fields will be added to this object upon conclusion of data mining activities.

The architecture flow then divides into two sub-systems of analysis; TF-IDF and Clustering, along with Doc2Vec Embeddings, before combining as a final dataset upon completion.

6.7.1 TF-IDF/Clustering/Topic Model

The TF-IDF/Clustering sub-system will support the bulk of the visualisation, and thus is the more complex of the two sub-systems, with a greater number of steps along the way along with a more complex output. It starts with a simplistic term frequency analysis, that just considers the commonality of terms across the corpus of ideas. This can be used to generate diagrams such as word clouds. This is extended by applying TF-IDF to the set of ideas to find the terms that are important to specific ideas but not common across the entire dataset.

8 https://www.mongodb.com/

TF-IDF values can then be converted to a matrix of features through vectorisation , and reduced to a two dimensional representation via PCA (Principal component analysis) (Wold et al. 1987). PCA aims to reduce the dimensionality of the data, while retaining how the data varies.

The sklearn package has a PCA decomposition method included to enable this. This allows us to plot ideas in a two dimensional space using a scatter plot, as each idea feature will have both an X and Y plot value, which are attached to the normalised idea documents in the next step. This provides a tactile way to visualise the textual difference between ideas, as scatter plots are commonly recognisable formats.

The other direction in the flow chart following Term Frequency – Inverse Document Frequency is the cluster analysis. The Python k-means model can be fit to the TF-IDF matrix, which will use the pre-defined number of clusters to search the data and categorise it into k groups by the Euclidean distance from the cluster. The centroids of each given cluster can also be reduced via PCA decomposition to plot the centroids along with the ideas, and the cluster each idea belongs to is attached to the relevant document.

The ideas in each cluster are applied to the sklearn LDA (Latent Dirichlet Allocation) function, which produces a topic model analysis of the set of ideas from each cluster. The features are then attached to each cluster's document, to demonstrate the unseen themes from each cluster.

6.7.2 Doc2Vec Closeness

The other sub-system is the training and subsequent data revelations from a doc2vec model. Gensim provide a simplistic Python API for using a doc2vec model which will be used in this project (Rehurek and Sojka 2011). The idea corpus is pre-processed to create tokenised training data, along with tagged documents for the analysis. Tagged documents allow relating the document vectors to the original ideas using their unique _id field. The genism doc2vec will build a model with given parameters, build the vocabulary using the training corpus then test over a given number of epochs.

Once any given idea is chosen, gensim can use the trained model to infer a vector from the respective idea's tokens. This vector can be compared to the other vectors from the set of ideas, and the closest vector can be referred back to the original idea using the tag from the vector. This enables a production of an ordered array of ideas, from most to least similar in content. Suggestions and soft "duplicates" can be inferred from this information.

6.7.3 Post Analysis

Once the data mining activities have all been performed, the data will be packaged in a consistent manner that the visualisation will reliably be able to reason with. The ideas will be packaged into another JSON file that will respond to the initial HTTP POST request sent by the client.

The following is an example of the extra fields that will be added post-analysis:

- **Content:** Amalgamated field of all the text fields from the original idea
- **Cluster:** Identifier for the cluster the idea belongs to
- **Tokens:** Array of strings of each word from the **Content** field, delimited by spaces, with punctuation removed and caps normalised

- **Stems:** Stemmed copy of the **Tokens** array
- **TF-IDF Instance:** Coordinates for 2D TF-IDF matrix representation
- **Topic Modelling**: Data mined unseen themes from the idea content

6.7.4 Technology

The architecture of the backend will use Python for its widespread support of popular machine learning, with the Flask library to allow the analysis option to operate as a HTTP server. This architecture also mirrors the approach used in the existing Simply Do Ideas product stack, where the user interface also interacts with, and receives information from, a Python/Flask server.

In regard to future work, the project developed in this way improves its flexibility to be integrated into the existing product stack. This familiarity also aids the author with the speed and efficiency they can program the project. Experience in the technologies used will incur less blockages related to programming ability, so more time can be spent completing "useful" work, rather than learning other technologies.

Python has a number of popular, simple APIs for conducting exploratory data analysis, making it a powerful and versatile tool for analysis. Both scikit (Pedregosa et al. 2011) and gensim (Rehurek and Sojka 2011) have well-documented implementations of the data mining activities decided for the project, that offer simplistic black box approaches to these processes.

An often discussed programming paradigm is to not "reinvent the wheel" (Gama 2007), to make use of the wealth of peer-reviewed existing implementations for tools rather than design it yourself. This project will make use of that approach to maximise reliability, success, and speed of the development process. This project does not aim to provide a confirmatory analysis of the effectiveness/speed of different algorithms.

A HTTP Flask server also aids the black-box approach to developing the analysis system. A user interface can send data to the server using HTTP requests, and will receive analysed data in return, without needing any understand of how the analysis is performed. Python Flask also offers a range of features that aid the fundamental development process. Errors are handled without causing a server crash, and changes to the code are applied through automatic restart on save. Once the user interface begins development, testing effectiveness can be done rapidly by sending and receiving data to the server.

6.8 Static Architecture (Visualisation)

The following diagram, Figure 9, demonstrates how the output from the analysis system could be visualised. The coloured circles in the image are representations of ideas, with a colour to indicate which cluster it belongs to. Figure 10 provides a real example of how an idea looks within a list of ideas, and how they will be displayed in the final iteration of the project.

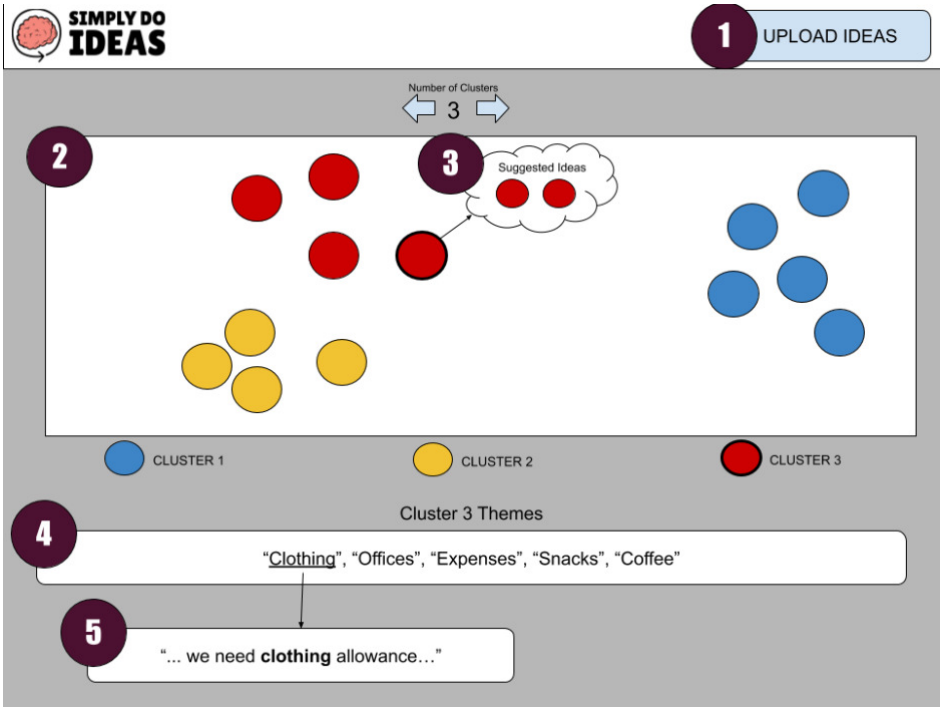

Figure 9 - User interface design

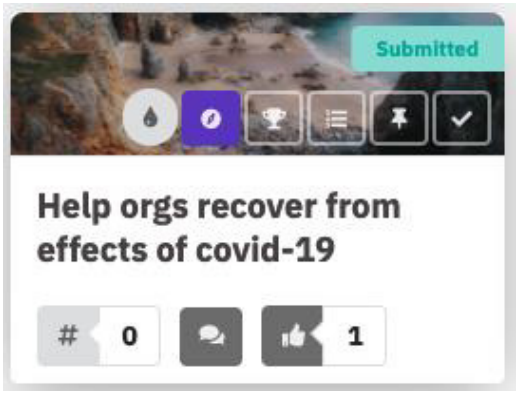

Figure 10 - Example of an idea "card" in Simply Do Ideas application

The diagram Figure 9 is taken from the perspective of the end user – typically the challenge owner or the consulting staff member. The visual elements are indicative of the actions user will be able to take, along with the visualisation concepts that can be expected, specific design of individual elements is subject to iterative change and evaluation.

As a companion Figure 11 contains examples of how the post ideation process can work using the project, and how a user reduces the cognitive load of managing a challenge.

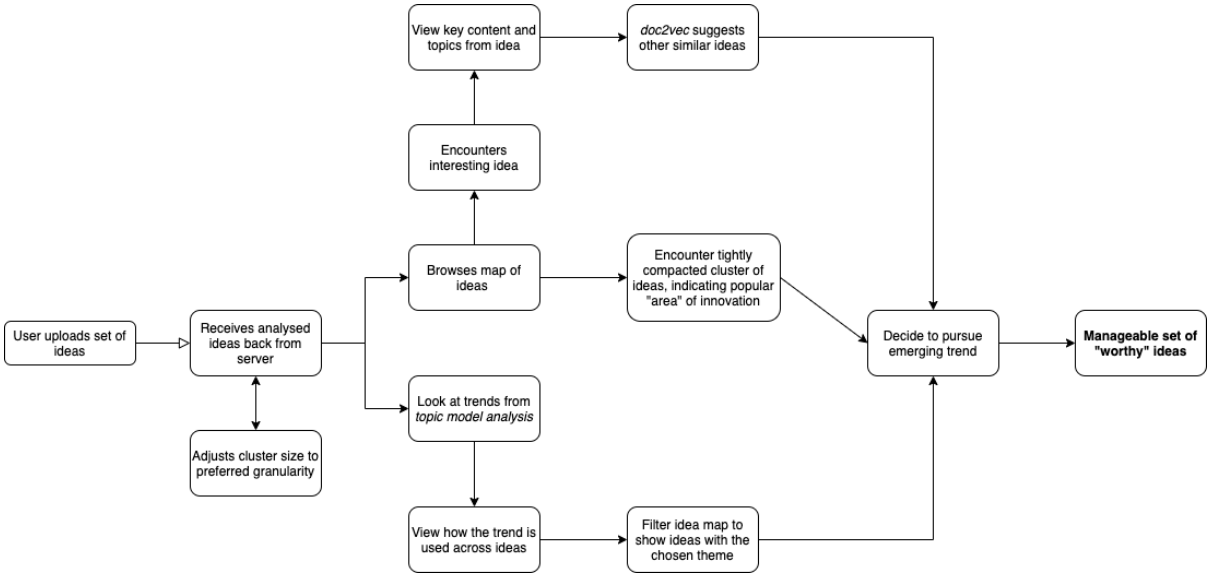

Figure 11 - Visualisation user flow

The first step in the user flow involves them uploading a set of ideas that they have acquired from a challenge. This step is unique to the project, and assuming the system would be integrated with the existing product stack, would be no longer required as ideas would be manually imported.

For the scope of the project it is necessary to allow the system to act independently. This action is performed using Action 1 in Figure 9. Once the server has analysed the ideas, they will be returned to the client to be visualised. As shown in Figure 11, the client may adjust the number of clusters, which will subsequently return the ideas to the database to be reanalysed with the new k value for the k-means algorithm.

The following steps indicate paths the user could take in the visualisation following receiving the analysed idea data back, either “Browses map of ideas” or “Look at trends from topic model analysis”. These paths are not mutually exclusive, and optionally both can be undertaken. This allows the challenge owner to deep dive into the analysis if they choose to, but can pick a path exclusively if they are trying to save as much time as possible.

6.8.1 Trend Analysis

Taking the path of topic model analysis, the user will begin on the user interface using Action 4. This will contain the trends uncovered from the dataset as a whole, or a specific cluster if desired. The “themes” will be a discrete set of single words that have been uncovered as popular among the data. In the example these include “clothing” and “offices”. Stop words are excluded from analysis to prevent connectives from being included in the themes. These words will enable the challenge owner to instantly uncover the most popular trends among the ideas without having to take any additional actions.

Hovering over any given word transitions the user into the next part of the flow, “View how the trend is used across ideas”. Hovering over the word will reveal context of how the word is used across the ideas that contain it.

The system will identify usages of the word across all ideas, then attempt to provide information on how that is used, by choosing words around it to add meaning, depending on where the word lies within an idea’s content. If the system provides a 16-word context for any

given word, then these words are chosen based on the given word's location. If the given word is at the beginning of the idea, the subsequent 16 words will be used. If the word is in the middle of an idea, the preceding 8 words and the following 8 words will be used. An example of this using the given word "office" may be:

- "…. We feel that when we are in the **office** needs more water coolers to give us water…"

Internal challenge report templates use this format for visualising trends. When a subject matter expert produces a post-challenge document describing the themes, they use this method of providing context. This feature aims to emulate that functionality to retain the "natural" feeling.

During the planning of the project, it was intended that the user could only view topics for a specific cluster, but following evaluation with the Simply Do Ideas team, this was extended to show topics for the entire set of ideas – allowing analysis as soon as the data is received from the server.

The final step in this branch of the user flow is filtering ideas to only those that include the given word. By progressing from hovering over the word "office", to clicking on it, the map of ideas is filtered to only display those with the word "office" within their content. This reduces the visual complexity of the scatter plot and empowers the challenge administrator to further investigate the intricacies of the trend.

6.8.2 Cluster Analysis

The second branch following from receiving the analysed ideas from the server involves directly browsing the scatter of ideas, beginning with Action 2 on the user interface in Figure 9. The scatter plot has a number of controls and keyboard shortcuts that improve the browsing experience. The user may pan the map in any direction, as well as zoom in or out. This function provides the user with a tactile, exploratory method to explore the analysed idea based on how close their content is. The user can naturally explore any trends and identify groups of ideas that may overlap, or ideas that are very different than others if they are placed at a distance from any groups. As k-means cluster analysis assigns every observation to a cluster, it does not identify ideas that may be outliers from the norm.

It is out of the scope of the current project to use a complex avant-garde solution such as cluster outlier detection (Duan et al. 2009), so to alleviate this the map will allow visual identification of ideas that may present unique and interesting content. It will additionally highlight very close or duplicate ideas, if a group of them are heavily overlapping.

This branch of the user flow supports the use-case where the user wishes to be more involved in the process. Some existing challenge owners across organisations in the Simply Do Ideas platform prefer to produce their own reports, unassisted, and investigate trends themselves. The idea map is carefully designed to support, rather than takeover, this process. It makes suggestions to the user but does not force them to take any activity, allowing them to take advantage of the richness of the analysis available.

The idea map will also offer a "viewed ideas" option – Users will be able to hide ideas that they have already viewed, so they can whittle down the dataset through their exploration.

6.8.3 Managed Cluster Analysis

Using the idea map will also support a semi-managed user flow. This approach will support users who want a middle ground between relying entirely on the automatically analysed trends, and sifting through the map manually. The user interface provides controls to reduce the cognitive load by limiting ideas to those of a specific cluster. By clicking on a cluster, they will only see the ideas and themes for that cluster. The user can then browse ideas from only that cluster. When a user sees an interesting idea, hovering over it will display "key content" – Displaying content from the idea that contains key terms from the cluster that it belongs to, using the previously mentioned context identification. Additionally, as seen in Action 3 on the user interface, the user can make use of the doc2vec vector similarity to find "Suggested Ideas" based on one that interests them, which is a popular approach among recommendation systems across the internet (Nandi et al. 2018). They can then choose to manually investigate these ideas.

6.8.4 Post Analysis

These three processes all converge on the final two phases, "Decide to pursue emerging trend" and "Manageable set of worthy ideas". These two phases are out of the scope for the current duration of the project. They work on the assumption that the project has met the success criteria laid out, and allowed the user to find a useful set of ideas using the system and has chosen to act on them in the real world; make changes to the office, improve their logistics, or whatever the respective proposed outcome from the challenge process was intended.

The Simply Do Ideas platform already offers a system for managing the real-world implementation of ideas using a Kanban (Anderson 2010) (Figure 12) approach, and a successful inclusion of this project in the product stack would slot in before this in the challenge process.

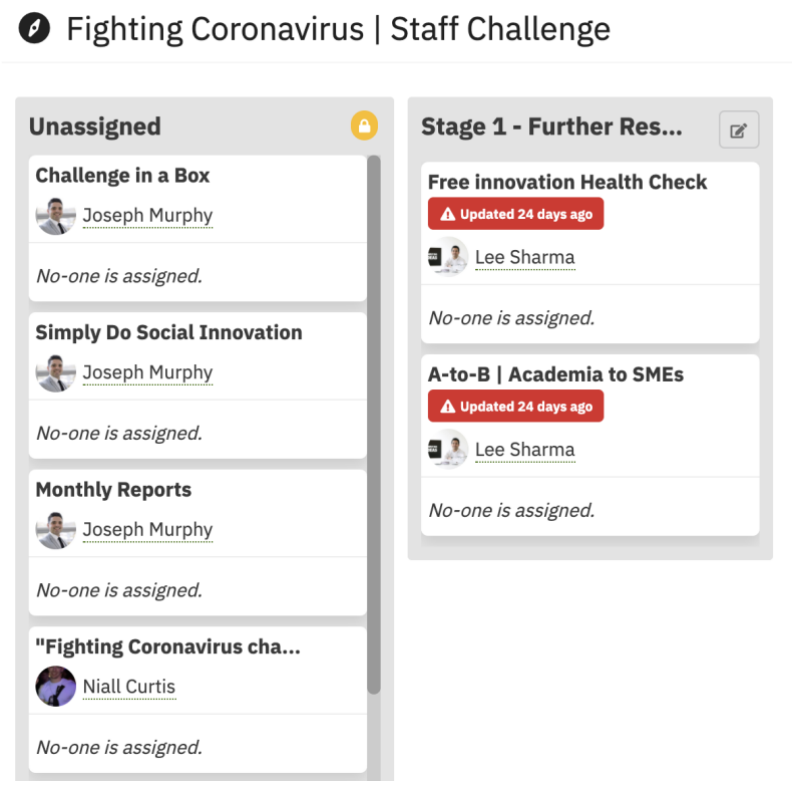

Figure 12 - Idea Kanban board

6.8.5 Technology

The architecture of the visualisation is all built on ReactJS, currently the most popular platform for building responsive front-end web applications (Robbestad 2016). That platform is also in line with the existing Simply Do Ideas product stack and is an area of experience for the author, streamlining the development of the basic user interface and allowing more time to be spent on the intricacies of data visualisation.

This will be supported by the semantic ui visual design library, which adds a number of common UX paradigms as simple components – Buttons, Inputs, Headers can be added with simple single components. These two libraries mean the surrounding visualisation for supporting the data visualisation can be developed smoothly and quickly.

React uses a component lifecycle model to enable responsive user interfaces, which can react to changes from an API. Once the user uploads a set of ideas from the user interface, the frontend will automatically update when the analysed data is received. This removes the need to refresh once ideas are analysed and is a friendlier experience for the user.

There is no popular standard for data visualisation in React. Many of the React implementations are based on d3.js, a low-level data visualisation library that works across different JavaScript implementations (Zhu 2013). The author researched a number of potential candidates for rendering a scatter plot in React. These included ChartJS[9], React-Vis[10] and Victory[11] (Iglesias 2019). Eventually the author settled on using NIVO – This was due to the extensiveness of documentation, flexibility of customising their scatter plot implementation, simplicity to use and attractive visual design.

9 https://www.chartjs.org/
10 https://uber.github.io/react-vis/
11 https://formidable.com/open-source/victory/

7 Implementation

This section will detail, to the code level, how the specifications detailed in the previous section have been implemented in the project. It will also detail how the final project differed from the original design; systems that were changed due to self/stakeholder evaluation, limitations due to complexity or unforeseen complications, and areas that may have been over ambitious. It will follow the same structure as the Specification & Design section for coherent reading.

7.1 Analysis Backend

7.1.1 Data Clean-up

Cleaning up the received ideas is one of the tasks when developing that took more time than expected. Even though it only represents one of the minimum requirements from the stated requirements, it required a greater-than-expected level of work. The following figure is an example of an idea before it has been normalised by the system. The clean-up tools are located in "*analysis/tools/Cleanup.py*". Ideas are initially imported into a Python list by using Python's in built JSON handling.


```
_id: "5c643561fc0d3f00078d02c9"
challenge: "5c3c9c9809def7000170bca6"
createdAt: "2019-02-13T15:18:57.065000"
isPinned: true
isSubmitted: true
name: "Competency-related threshold pay"
▶ projectManagement: {5c3c9c9809def7000170bca6: {…}}
▶ stamps: ["green"]
▶ storedMedia: {uploadcare: Array(5)}
submittedAt: "2019-02-13T15:40:54.455000"
template: "5c3c9c9809def7000170bca5"
▼ templated:
  ▼ 1d8a816d-aa3c-4ff5-ae27-3289784efc52:
      notes: "I believe this would not be that difficult to implement
      value: 8
    ▶ __proto__: Object
  ▼ 50156801-b83c-4395-8d2e-30ff1807f2c4:
      notes: "Very Important, you work very hard to become a Detective
      value: 10
    ▶ __proto__: Object
  ▼ 85fd9fc6-ed3f-4454-bf35-b0771039c7f1:
      notes: "If this was implemented I honestly feel that officers wo
      value: 10
    ▶ __proto__: Object
    f196a2c8-fb78-41d4-b752-82742e07abd5: " CRTP - Detectives we first
  ▶ __proto__: Object
updatedAt: "2019-02-13T15:40:25.298000"
user: "5c643498fc0d3f00078d02c1"
voteCount: 13
```


Figure 13 - Un-normalised idea input

Many of the fields here are unnecessary for the analysis; the only fields being analysed within the scope of this project are the content fields, which are contained under the key templated. These are user inputs from text input fields in the idea's "templated", the format in which the challenge owner has specified they format their ideas. The data mining NLP tools being used in the project require the inputs to be single strings. Another challenge is that the format of the templated field varies from challenge to challenge dependent on template, which requires the clean-up function to be fairly robust in how it handles the documents.

```
# Iterate recursively through idea until string is found and concatenate content
def untilString(data):
  if isinstance(data, str):
    return(data)
  elif isinstance(data, dict):
    if data.get('notes'):
      untilString(data.get('notes'))
    elif data.get('value'):
      untilString(data.get('value'))
```

Figure 14 - Function to reduce templated field to single string

Figure 14 demonstrates a function that can reduce the templated field to a single string. It takes the templated field, or any object, as an input, and recursively iterates until it reaches a field called either notes or value. These are guaranteed to be string content fields, which are rules of the Simply Do Ideas platform. By applying it iteratively to all the fields within templated, as seen in Figure 15 (as well as encoding and decoding into ascii format to rid any erroneous characters), the system can convert all of the idea's analysable content into a single string.

```
ideaContent = ''
for k, v in idea.get('templated', {}).items():
  if isinstance(v, str):
    v = v.encode('ascii', 'ignore').decode('ascii')
    v = str(v)
    ideaContent += "{0} ".format(untilString(v)).replace("\n", "")
```

Figure 15 - Applying untilString to an idea's content

This function is used as a greater whole of a function called normaliseIdea (Figure 16).

```
def normaliseIdea(idea):
  # Normalised idea format for analysis
  ideaForAnalysis = {
    '_id': str(idea.get('_id')),
    'name': str(idea.get('name', 'Untitled idea')),
    'content': '',
    'stems': []
  }
  ideaContent = ''
  for k, v in idea.get('templated', {}).items():
    if isinstance(v, str):
      v = v.encode('ascii', 'ignore').decode('ascii')
      v = str(v)
      ideaContent += "{0} ".format(untilString(v)).replace("\n", "")
  soup = BeautifulSoup(ideaContent, 'html.parser')
  html_content = soup.get_text()
  ideaForAnalysis['content'] = html_content
  return ideaForAnalysis
```

Figure 16 - Normalise idea function

This function sets out the standardised format for ideas to be analysed upon. It additionally uses an extra Python library called Beautiful Soup[12] (Richardson 2007), which provides an API to pull content from HTML. Some ideas in Simply Do Ideas have a rich text input, where users

[12] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

can format their content with indentation, bold, underline, images and more. This is then stored as HTML in the server, and parsed on the web application. This adds unwanted extras to the idea content for analysis, as the HTML tags and punctuation are not analytically valuable. Beautiful Soup uses a Regular Expression approach to pull the "content" from a HTML string, the writing of the user.

Once the entire corpus of ideas has been normalised into the same format, the content is further broken into the tokens and stems which will be used for the Natural Language Processing.

```
def tokenizeAndStemIdeas(ideas):
    stopwords = getStopwords()
    stemmer = SnowballStemmer("english")
    # Use tweet tokenizer as idea content can be variable in grammar and correctness
    tokenizer = TweetTokenizer()
    for idea in ideas:
        ideaContent = idea.get('content')
        # Tokenize string into individual words
        tokens = tokenizer.tokenize(ideaContent)
        # Filter any strings not containing letters
        filtered_tokens = []
        for token in tokens:
            if re.search('[a-zA-Z]', token) and token.lower() not in stopwords:
                filtered_tokens.append(token.lower())
        # Stem words
        stems = [stemmer.stem(token) for token in filtered_tokens]
        idea['stems'] = stems
        idea['tokens'] = filtered_tokens
    return ideas
```

Figure 17 - Tokenizing/Stemming Ideas

This function uses the TweetTokenizer from Python's Natural Language Toolkit NLTK (Loper and Bird 2002), which is a robust tokenizer intended for usage where correct grammar and English cannot be guaranteed, and expect usage of emoticons or spelling errors; built for usage on tweets from Twitter, but applicable anywhere that receives unmoderated user input.

Tokens are then filtered against a pre-determined list of analytically "useless" words that are common across ideas that are different in content. As well as standard stop words from NLTK, this project adds the most commonly used words across **all** ideas in Simply Do Ideas, regardless of challenge. A list of all stop words is contained in the supplementary materials, and include subject specific terms such as: "idea", "save", "youtube", "vimeo" – the latter of which are picked up by NLP tools when ideas contain links to external videos.

These tokens are then stemmed using *NLTK's SnowballStemmer*. This function stems the filtered tokens using Snowball, a language built for stemming algorithms. More information on the stemming algorithm used can be found on pages 130 through 137 of Porter's "An Algorithm for suffix stripping" (Porter 2001).

Both the tokens and stems are affixed to the previously created normalised idea document, and the corpus is returned to proceed to further analysis.

7.1.1.1 Constraints

While robust, there is still further complications when cleaning up an idea document. The normalisation activities performed do not always guarantee that an idea will be useful for analysis, as there are a number of uncertainties. The following list demonstrates limitations in the approach:

- **Spelling:** The platform nor the server have any guidelines or handling for the misspelling of words. Key theme and nouns can be entirely missed if misspelled throughout the content of an idea. For example, an idea where “office” is continually spelled as “ofice” would be entirely different in the eyes of the analysis as one that uses the word “office”.
- **Grammar:** Similarly to spelling, the platform makes no attempt to alleviate grammatical errors. Words that directly precede or follow punctuation may be incorrectly picked up.
- **Unfamiliar terminology:** While it is robust in handling many words, there is always the potential for unfamiliar terms to slip through and cause issues analytically. Encoded images, unusual link formats and more.

7.1.2 Server Model

The server uses a very simple, limited set of endpoints for its requests. When the direction of the project pivoted to do analysis via a server model, the system simply had Flask added to base and the functions were extended to be called upon when different requests were made.

```
@app.route('/analyse', methods=['POST'])
def analyse_ideas():
  if request.method == 'POST':
    req_data = request.get_json(force = True)
    ideas = req_data.get('ideas', [])
    options = req_data.get('options', {})
    return jsonify(IdeaAnalysis.analyseIdeas(ideas, options))

@app.route('/doc_to_vec', methods=['POST'])
def doc_to_vec():
  if request.method == 'POST':
    req_data = request.get_json(force = True)
    ideas = req_data.get('ideas', [])
    options = req_data.get('options', {})
    return jsonify(IdeaAnalysis.doc2vec(ideas, options))

@app.route('/closest_ideas', methods=['POST'])
def closest_ideas():
  if request.method == 'POST':
    req_data = request.get_json(force=True)
    idea = req_data.get('idea')
    ideas = req_data.get('ideas')
    return jsonify(IdeaAnalysis.closestIdeas(idea, ideas))

if __name__ == '__main__':
  app.run(debug = True)
```

Figure 18 - Analysis Server Endpoints

The server only accepts three endpoints: analyse, doc_to_vec, and closest_ideas. The analyse endpoint returns to the client a set of analysed ideas using the main static flow of the analysis system – TF-IDF, clusters and topics. It provides all the necessary information for the visualisation to render the idea map and the top themes.

The doc_to_vec end point returns a PCA reduced two-dimensional representation of the vector distance between every idea in the set. This end point ended up being excluded from the visualisation due to changes in the vision of the project and time limitations as the project neared conclusion.

The closest_ideas end point accepts a set of analysed ideas along with an individual idea, which has its vector compared to those of the entire idea set in order to use doc2vec for suggesting similar ideas. This end point can only currently be called if the visualisation has previously received analysed ideas.

7.1.3 TF-IDF

Following the clean-up of the dataset, the normalised ideas can be fed into the analysis functions. The implementations for TF-IDF and Clustering are in analysis/tools/BasicAnalysis.py. This computation begins with transposing the content from every idea object into a single array of content strings, which is then usable by the data mining libraries gensim and skicit. The TF-IDF in this project's analysis makes use of the TfidfVectorizer in skicit. This function is a combination of converting a corpus of text to a count matrix, following by fitting that to a tf-idf matrix. The TfidfVectorizer is constructed using a parameter that specifies stop words to be ignored during analysis.

```
# Find tf_idf_matrix
vrz = TfidfVectorizer(stop_words=stopwords)
tf_idf_matrix = vrz.fit_transform(all_idea_contents)
```

Figure 19 - TF-IDF Analysis of ideas

More specifically, TfidfVectorizer combines two other functions from scikit: CountVectorizer and TfidfTransformer. CounterVectorizer simply produces a term frequency calculation for all of the supplied strings. For example, given a sample of:

- ["simply do ideas", "ideas man", "woman simply"]

Feeding this array into the CountVectorizer would produce a representation of the following matrix.

	do	**ideas**	**man**	**simply**	**woman**
0	1	1	0	1	0
1	0	1	1	0	0
2	0	0	0	1	1

Figure 20 - Example Word Frequency Matrix

Without providing additional parameters, this function delimits the content by space (or uses a provided array of tokens) then counts the occurrence of each word in the dataset across every document, producing a matrix for the entire corpus. Following generation of this matrix, the TfidfVectorizer then transforms this count matrix into a normalised tf-idf representation. The documentation for scikit states that it uses the following formula for calculating the tf-idf for term t in document d of a document set (Pedregosa et al. 2011): "tf-idf(t, d) = tf(t, d) * idf(t), and the idf is computed as idf(t) = log [n / df(t)] + 1", where n is the total number of documents in the set and df(t) is the document frequency of the original t. Applying this to the example count matrix would produce the following corresponding tf-idf matrix, which now applies each word a normalised value based on its importance to the overall corpus as well as just individual documents.

	do	**ideas**	**man**	**simply**	**woman**
0	0.680919	0.517856	0.000000	0.517856	0.000000
1	0.000000	0.605349	0.795961	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.605349	0.795961

Figure 21 - Example TF-IDF matrix

This matrix then has a two-fold statistical significance. It will both enable future k-means analysis, and be reduced from a three dimensional matrix to a two dimensional representation for visualisation plotting.

7.1.4 Clustering

Upon completion of the tf-idf matrix, the data now has the appropriate attributes to be clustered using k-means. The scikit implementation of k-means, by default, solves the problem using an implementation of *Lloyd's algorithm*, also known as Voronoi iteration. The algorithm uses Euclidean spaces subsets and repeatedly finds the centroids for each partition set, clustering on the closest centroid for each observation (Lloyd 1982). The average complexity of scikit k-means is O(K n T), with n samples and T iterations.

```
# Random value of K for K-means
k_value = options.get('numberOfClusters', 3)
cluster_model = KMeans(n_clusters=k_value, init='k-means++')
labels = cluster_model.fit_predict(tf_idf_matrix)
```

Figure 22 - Scikit K-Means Implementation

Figure 22 shows how the project implements scikit k-means. The number of clusters is specified by the user in the visualisation, but arbitrarily defaults to three if it is not specified. The model is initialised, then is fit and predicted to the tf-idf matrix. Fit_predict is a convenience function that runs fit and predict from scikit in succession. Initialising the model with the k-means++ parameter ensures it uses "smart" random centroid distribution, whereby the prediction will attempt to avoid choosing centroids that are too close together.

Figure 23, taken from the paper "Metamorphic Exploration of an Unsupervised Clustering Program" (Yang et al. 2019), visualises the process of fit then prediction, where fit is the computation of the k-means clustering centres, and predict computes the closest cluster to which each document in the matrix belongs to.

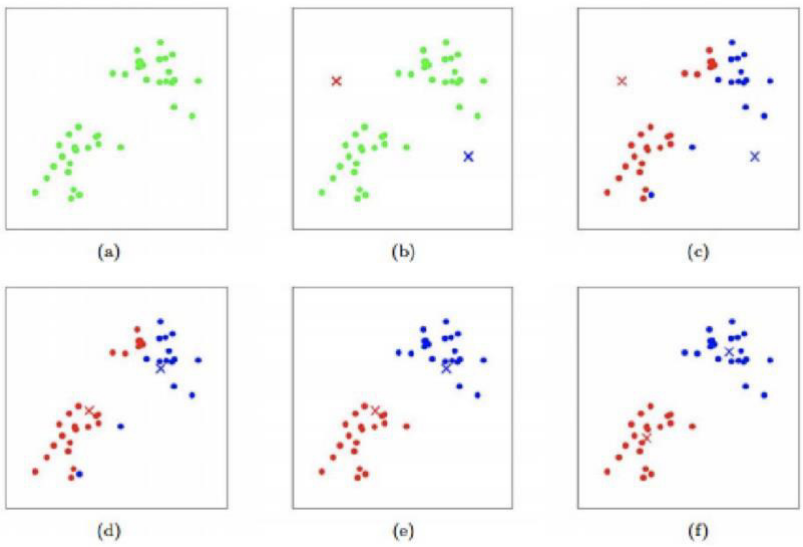

Figure 23 - Visualisation of clustering fit and prediction

The project proceeds to mine the knowledge uncovered from the k-means analysis.

```
centroids = cluster_model.cluster_centers_
order_centroids = centroids.argsort()[:, ::-1]
terms = vrz.get_feature_names()
```

Figure 24 - Cluster infomation mining

The call to cluster_centers_ finds the coordinates for the centre of the k clusters (in this example, k is three). Given a sample of 50 observations and k equal to three, cluster_centers_ would be a matrix with k rows and 50 columns. Sorting these centroids using .argsort()[:, ::-1] is the method to find the most popular terms from the corpus. As columns is equivalent to words in the k-means matrix, sorting the the matrix in descending order uncovers the most popular words. Get_feature_names() is then a simple mapping function provided by scikit to relate the column index to its specific term rather than just an integer.

A list of cluster information is then built, with relevant data to allow visual manipulation of the center of the clusters. This includes:

- Index of the cluster
- Stems and terms from the cluster
- Coordinates for the centroid

The most popular terms found by the code in Figure 24 are mapped to the document to enable the *"low effort"* user flow from Figure 11.

7.1.5 Principal Component Analysis

The final step for the use case of visualising tf-idf and cluster modelling is to convert the data into a format that is easily represented in a web application, where three dimensional plots are not feasible. This is done using Principal Component Analysis (PCA) (Wold et al. 1987). In Layman's Terms, PCA is a technical implementation of explaining something simply, by reducing the dimensionality. It uses best effort to visualise the relationship between observations in a three-dimensional space, with a two dimensional representation, based on variance between observations.

In this project, PCA is implemented to visualise how close ideas are in a relative sense, where the exact values are not relevant for the end user. The most important thing for visualisation is that ideas that are very different from each other are suitably distant when the dimensionality is reduced to two dimensions, while the precise distance between them is both unneeded and not displayed to them.

```
pca_num_components = 2

dense = tf_idf_matrix.todense()

# Reduce 3 dimensional tf_idf_matrix to a 2 dimension representation for plotting
pca_model = PCA(n_components=pca_num_components)
reduced_data = pca_model.fit_transform(dense)
for index, instance in enumerate(reduced_data):
  ideas[index]['tf_idf_instance'] = {
    'x': instance[0],
    'y': instance[1]
  }
  ideas[index]['cluster'] = int(labels[index])

# Get 2 dimensional representation for cluster centroids
centroids_reduced = pca_model.transform(centroids)
centroid_plots = []
for index, instance in enumerate(centroids_reduced):
  centroid_plots.append({
    'x': instance[0],
    'y': instance[1]
  })
```

Figure 25 - PCA on idea matrices

We begin in Figure 25 by specifying that we want PCA to reduce to two components, and converting the tf_idf_matrix to a dense copy. The algorithm requires all values to be stored, including non-zeroes, which necessitates a dense representation. We then construct the model using the decided number of components using PCA(), and both fit the matrix and apply dimensionality reduction using fit_transform. Each idea in the corpus is given it's coordinates from this reduced data, and the cluster it belongs to. The same process is also applied to the centroids of the clusters to enable visualisation of them.

The final step of PCA represents complete coverage of the **minimum** functional requirements for the analysis system. These steps were all completed at the start of the development process to ensure it could achieve its required functionality, and the **desirable** steps were included afterwards. Some **desirable** functionality is interspersed among the code stated above, and the figures represent important snippets of analysis but are not an all-encompassing guide to the codebase. Following this, the author implemented code to achieve the **desirable** requirement of topic modelling to further the richness and usefulness of the project's visualisation, both via LDA and doc2vec implementations.

7.1.6 Latent Dirichlet Allocation

The first approach to topic model analysis used is Latent Dirichlet Allocation.

```
def topicModelling(ideas):
  getContents = getAllIdeaContents(ideas)
  all_idea_contents = getContents['all_idea_contents_tokens']

  stopwords = getStopwords()

  vrz = TfidfVectorizer(stop_words=stopwords)
  tf_idf_matrix = vrz.fit_transform(all_idea_contents)

  LDA = LatentDirichletAllocation(n_components=1)
  LDA.fit(tf_idf_matrix)

  for i,topic in enumerate(LDA.components_):
    return [vrz.get_feature_names()[i] for i in topic.argsort()[-10:]]
```

Figure 26 - LDA for Ideas

Figure 26 demonstrates the project's black box implementation for LDA, which can be given any set of ideas and it will return a set of uncovered, data mined themes. The function begins with running the getContents function, which is a simple clean-up tool to get all of the contents arrays from each idea in the set and append them into a single large list of lists. topicModelling then uses the same method of tf-idf vectorization as discussed previously. The LDA model is then initialised.

An unfortunate consequence of using topic modelling in this context is that the sets of ideas are all relatively small size. The exploratory nature of topic modelling means that small datasets are subject to high levels of variance with diminishing accuracy. An article by Jason Brownlee (Brownlee 2019) discussing dataset sizes during exploratory machine learning has a high variance on accuracy dependent on dataset size. A typical array of ideas from Simply Do Ideas has around 100 ideas. The article discusses that data mining with 100 samples, evaluated against that of 100,000 samples, has a test accuracy of 72.041%. This is over 10% lower than the observed 84.025% accuracy when evaluating a sample of 10000 items. This implies that a certain level of unreliability will be observed when doing topic model analysis on ideas, especially on the level of individual clusters. An attempt to mitigate this is by specifying the LDA to only use one component, and find the top words within that topic. This is opposed to finding distinct topics within a dataset, then the words within them. The LDA model is fit to the tf_idf_matrix, and the same enumeration of the columns from earlier is performed to extract the top terms in descending order.

This function is called on both the full dataset, and the ideas within each cluster, allowing a varying granularity of analysis. If the user chooses to increase the number of clusters they will further increase how precise the themes are to a specific area of interest; however the important limitation is that for every increase in k, the accuracy of the topic model analysis continually decreases.

7.1.7 Doc2vec

The second implementation of topic modelling is performed via doc2vec, using an implementation from the gensim library (Rehurek and Sojka 2011). The doc2vec theory has

been discussed earlier in the report, this section will focus on usage of the model. The advantage of the doc2vec model over LDA is that it retains the ordering of the word in its analysis of term importance, as ordering is indicative of meaning in sentences.

```
def preprocessEmbeddings(ideas, tokens_only=False):
  data = []
  stopwords = getStopwords()
  for idea in ideas:
    if tokens_only: data.append(idea['tokens'])
    else: data.append(gensim.models.doc2vec.TaggedDocument(
      words=idea['tokens'],
      # Tag documents with _id so they can be retrieved from vector later
      tags=[idea['_id']]
    ))
  return data
```

Figure 27 - Pre processing for doc2vec

```
def doc2VecModel(ideas):
  train_corpus = preprocessEmbeddings(ideas)
  test_corpus = preprocessEmbeddings(ideas, tokens_only=True)

  model = gensim.models.doc2vec.Doc2Vec(vector_size=50, min_count=2, epochs=40)

  model.build_vocab(train_corpus)

  model.train(train_corpus, total_examples=model.corpus_count, epochs=model.epochs)

  return model
```

Figure 28 - doc2vec model processing

Figure 27 is an example of how the project pre processes a set of ideas for doc2vec topic modelling. It accepts ideas and a parameter tokens_only. This parameter should be specified when training a doc2vec model vocabulary, as the tags are not relevant at this part. Following this, the function now has a list of all the lists of tokens from each processed idea. The model is then created using the gensim doc2vec.

The important parameters supplied are min_count and epochs. A minimum count of two ignores words that are used less than twice, so are completely insignificant over a large dataset. The epochs indicate how many times the algorithm is iterated on the dataset. A study by Google indicates a typical iteration number for corpuses with up to millions of entries is 10-20 (Le and Mikolov 2014), so due to having a low number of documents in the dataset used here the author has opted to use 40 epochs. The model vocabulary is then built and trained on the pre-processed vocabulary, which now has a completed and trained model on the specified set of ideas, where the vector space between documents and words is now known. A visual example of such a trained set is visible in Figure 29 (Sadighpour 2016).

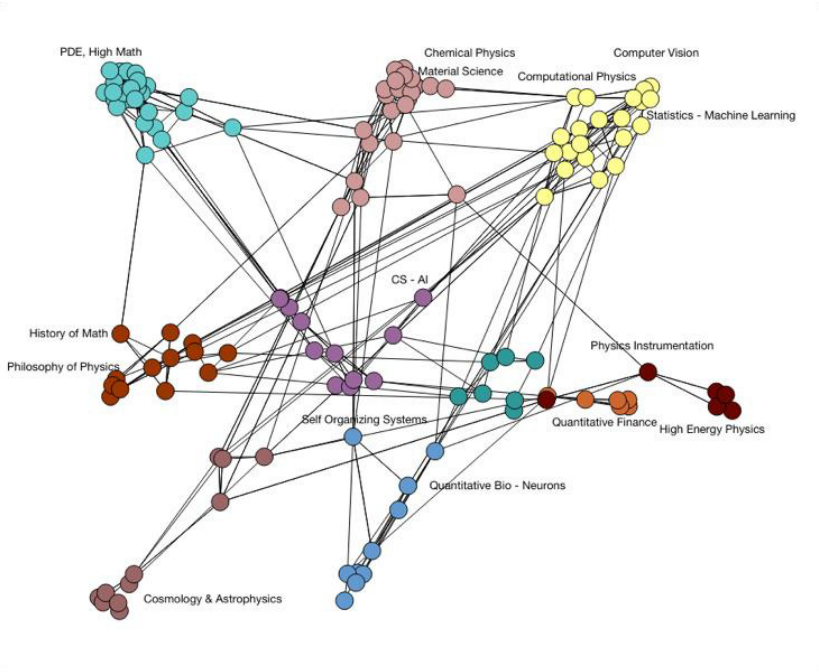

Figure 29 - Example of visualised, trained doc2vec

The model constructed in Figure 28 can then be leveraged to discover similar ideas to any given one. The function in Figure 30 can be called from the close_ideas endpoint in the API, as this information is accessed separately than the regular idea analysis functions.

```
def closeIdeas(idea, ideas):
  # Finding the closest ideas to a given one using a Doc2Vec embeddings model

  # Train model on existing ideas
  model = doc2VecModel(ideas)

  # Infer a vector for the selected idea
  idea_vec = model.infer_vector(idea['tokens'])

  # Get most similar ideas to idea using new vector
  most_similar = model.docvecs.most_similar([idea_vec])
  most_similar_ideas = []
  for tag in most_similar:
    # Get most similar idea from original corpus by using the Tag from the most_similar, which is the idea _id. We tagged these when pre processing
    most_similar_idea = next((i for i in ideas if i['_id'] == tag[0] and i['_id'] != idea['_id']), None)
    if(most_similar_idea):
      most_similar_ideas.append(most_similar_idea)
  return {
    'mostSimilarIdeas': most_similar_ideas
  }
```

Figure 30 - Close ideas functions

This function accepts the parameters idea and ideas, where idea is a potentially unseen idea, and ideas is a pre-analysed set of ideas from the earlier analysis function. The model is then used to infer a vector of the given idea. This applies the doc2vec function to the idea, and uses cosine similarity comparison to find the closest vectors to it. A caveat here is that due to the doc2vec being exploratory, iterative approximation, repeated evaluation of an idea may return slightly different results. As the author tagged the vectors with each idea's unique _id during pre-processing, the system is then able to procure the respective idea for each vector. The function then returns an array of the most similar ideas to be displayed in the visualisation.

7.1.8 Limitations

One of the original requirements and discussion pieces was using doc2vec as another method of creating an idea map. This was a natural progression from using the model, as following doc2vec analysis each idea has a numerical vector representation that it can display.

The beginnings of such implementation can be found alongside the other topic model functions.

```
def docEmbeddings(ideas):
  model = doc2VecModel(ideas)

  kmeans_model = KMeans(n_clusters=3, init='k-means++', max_iter=100)
  X = kmeans_model.fit(model.docvecs.doctag_syn0)
  labels=kmeans_model.labels_.tolist()

  l = kmeans_model.fit_predict(model.docvecs.doctag_syn0)
  pca = PCA(n_components=2)
  reduced_data = pca.fit_transform(model.docvecs.doctag_syn0)

  for index, instance in enumerate(reduced_data):
    ideas[index]['doc_vector_instance'] = {
      'x': str(instance[0]),
      'y': str(instance[1])
    }

  return {
    'ideas': ideas
  }
```

Figure 31 - doc2vec two dimensional analysis

This function follows a similar formula to the k-means analysis, where the doc2vec vectors are reduced in dimensionality to be plotted on the web application. While this implementation on the analysis server is fully featured, an evaluation on the project's usability rendered this obsolete.

It was decided that having two separate ways to plot the idea map increased the complexity of the user experience too much, and did not offer enough variance from the k-means visualisation to make it worth including. An additional issue was that, due to the inappropriately small dataset size for doc2vec, the variance in plotting the results was high – repeated modelling produced very different scatters. As such, this function remains currently unused but could be utilised for comparison in the future, or if a more complex workflow was required.

7.2 Visualisation

Figures 32 and 33 are indicative of the final iteration of the visualisation, at time of writing. They demonstrate how the system will look to the end user following uploading of a set of ideas, with the entire dataset mapped onto a graph with the controls visible, along with the themes discovered through topic modelling, and the differentiation of the clusters by colour code.

The individual components in these screenshots will be discussed in the order a typical user may interact with them, with code discussed if necessary. The overall visualisation is based wholesale on the initial diagrams created when planning, and refined through iteration. The styling of components was built in-line with the existing visual identify of the Simply Do Ideas

platform. This styling is backed by an internal style guide, which was externally designed by a user experience consultant and verified by developers within the company.

The style guide is backed by scientific user experience principles on usability and visual identity of components, ensuring that decisions made with regards to visualisation are consistent and have valid reasoning. The flow of the visualisation ends once a user has uncovered trends of ideas, or single ideas, that they feel suitably complete the original challenge specification. Once the visualisation is implemented in the existing Simply Do Ideas platform, the user would move from the project to the "idea project board", where ideas are managed by their progress in being implemented in the real world. As such, the project does not attempt to provide any implementation for showing which ideas are chosen to be taken forward – doing so would inflate the scope, and reinvent the wheel.

The visual design of the project follows these guidelines, and this has been confirmed by demonstration with staff members in the company. Changes were made where necessary to adhere to this.

Figure 32 - Visualised map of ideas

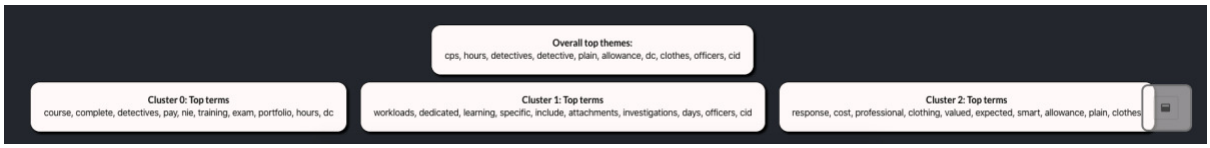

Figure 33 - Topic modelled "Top themes"

7.2.1 Uploading Ideas

As seen in the top right of Figure 32, the button is used for uploading the ideas for analysis. This button is backed by the JavaScript FileReader functionality (Cameron 2013) – which accepts a json type file, and then uses the JSON functionality to parse this data into an array of ideas. Every set of uploaded ideas is considered as a new analysis "flow", and therefore the entire state of the application is reset to allow the user to restart the activity.

```
this.fileReader = new FileReader();
this.fileReader.onload = (event) => {
  this.setState({
    originalIdeas: [],
    ideas: [],
    ideasAnalysedFormatted: [],
    error: false,
    clusters: [],
    clusterCentroids: [],
    termContextLoading: false,
    termContextIdeas: [],
    activeClusters: [],
    activeWord: '',
  }, () => {
    const { ideas } = JSON.parse(event.target.result);
    const ideasWithContent = ideas.filter(i => util.ideaHasContent(i))
    this.setState({ ideas: ideasWithContent, originalIdeas: ideasWithContent, error: false }, () => this.analyseIdeas());
  })
};
```

Figure 34 - Reading ideas from JSON

Once the ideas are read in, the ideas are sent to the analysis system using local API requests. As both the visualisation and analysis server were built to act independently, the visualisation does not need to know the progress of the analysis or need to understand anything that is happening "beneath the hood". Once ideas are uploaded to analysis the visualisation enters a loading state, then once data is received back it renders what it has received, with no underlying knowledge of what happened between those processes.

7.2.2 Adjusting the Granularity

When the ideas are uploaded to the server, a local copy of the chosen data set is saved, and it also includes a user specified number of clusters. The user can use the controls above the idea map to adjust the number of clusters.

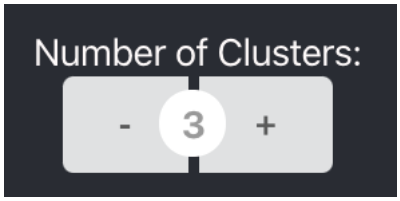

Figure 35 - Cluster adjustment

When the number is adjusted, the locally saved dataset is reuploaded to the server along with the new cluster number. The user is free to make as many adjustments as they wish, as well as leave the number of clusters at the default number. While three clusters is an arbitrary decision, many of the manual idea reports from the company combine ideas into three groups. This process can be repeated as many times as desired before moving forward in the user flow.

7.2.3 Browse map of ideas

The next proposed step for the user is to browse the returned map of ideas based on their tf-idf values. The user interface was designed for this to be the logical next step, by making it the prominent element on the screen. The map continues from directly below the controls. This is where nivo scatterplot (Benitte 2019) is used. The author has previously discussed the reasons why nivo is used, namely for its visual design and extensibility in customisation beyond the default design.

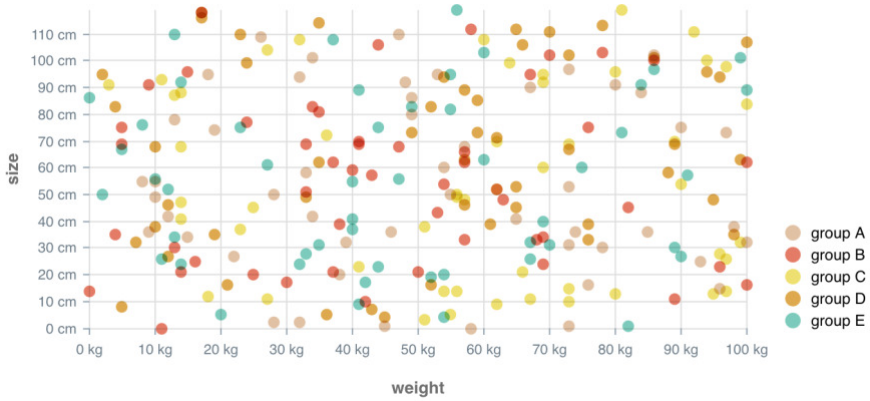

Figure 36 - Default nivo scatterplot

The visualisation uses a highly personalised version of nivo scatterplot with custom renderers. A caveat of using nivo is that it requires some clean-up on the frontend to format the data to one that is reasoned by a nivo scatterplot.

```
convertToChartData = (ideas, clusterCentroids ) => {
  const { numberOfClusters } = this.state;
  const ideasAnalysedFormatted = []
  for(let i = 0; i < numberOfClusters; i++) {
    ideasAnalysedFormatted.push({
      id: `Cluster ${i + 1}`,
      data: ideas.filter(idea => idea.cluster === i).map(idea => idea.tf_idf_instance)
    })
  }
  this.setState({
    ideasAnalysedFormatted: [...ideasAnalysedFormatted, { id: 'Centroids', data: clusterCentroids }]
  })
}
```

Figure 37 - Converting to nivo chart data

The visualisation uses the data returned from the analysis and creates three arrays, based on the number of clusters. Each cluster creates an object with an id based on the cluster number, and filters the ideas from the server to the relevant cluster, then pushes each idea's tf-idf value. In hindsight, much of this processing should've been done on the server – abstracting calculation from the visualisation. Processing information in this way on the front-end increases how coupled the server and interface are, reducing its black box possibility. Once the data is formatted, the ideas are displayed on the plot like in Figure 32.

An important part of the user being able to understand the map is by being able to manipulate what they can see. On first load, the ideas are all displayed on screen and tend to overlap in a way that makes it difficult to understand what is going on. This is why the user is provided with a suite of controls to explore the data and adjust what they can see on the screen.

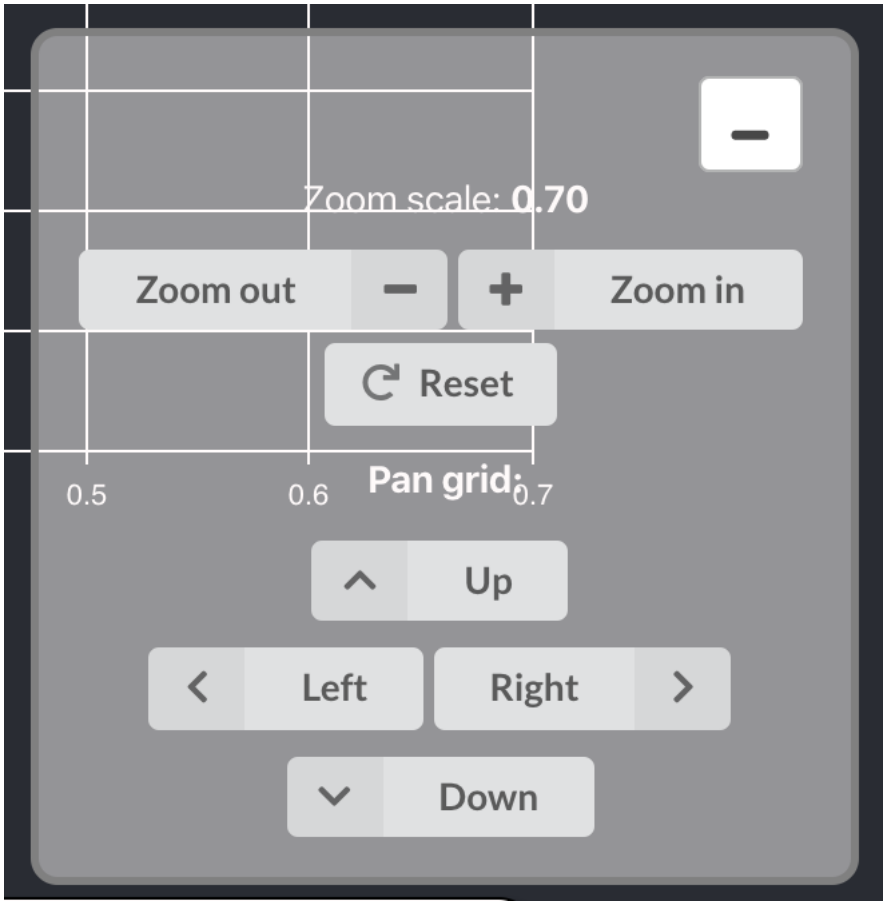

Figure 38 - Idea map controls

The user is given a multitude of controls to adjust what they see on their screen. The panning and zoom controls adjust the X and Y scale of the scatter, and the X and Y start/end values, respectively. For example, panning right one time would move the X lower bound ten points right, and the X upper bound ten points right. Figure 39 demonstrates how Figure 32 would look if the user zoomed and panned to get a more intimate understanding of the ideas in the dataset.

Figure 39 - Zoomed/panned idea map

The ideas are rendered using the bespoke idea card developed for the main Simply Do Ideas web app. This ensures the visual design closely resembles that of the Simply Do Ideas stack, and additionally reduces the burden of designing visual components exclusively for this project. Figure 39 also displays the differentiation of clusters by the coloured border placed on ideas. Two ideas that have "NIE" in the title are grouped in the same cluster, and close together on the idea map. The user is given a further set of controls to adjust what they can see on the map, if they still believe the cognitive load is too high.

Figure 40 - Further idea map controls

The controls shown in Figure 40 are fairly self-explanatory. The checkboxes, which automatically adjust based on the number of clusters selected by the user, allow them to decide which clusters they can see on the screen. The motivation behind these controls is that, assuming a user sees a group of ideas that they feel offer an interesting trend to investigate, they can limit their view to only those in that cluster.

The other control in Figure 40 is to hide "viewed" ideas. An idea is added to an array of viewed ideas once a user clicks on it to open it on the Simply Do Ideas application to view its full content. It will then add a green tick to the corner of the idea on the map, and if specified by the user, hide it entirely. This allows the user to "whittle down" the number of ideas they see on the map, as they begin to sift through them.

Once the idea map shows a limited number of ideas, the user can then begin to investigate individual ideas. Hovering over one of the idea cards shows the user "intelligent" context. This is parts of the idea's content that contains any of the topic model analysed themes, or terms from the title of the idea.

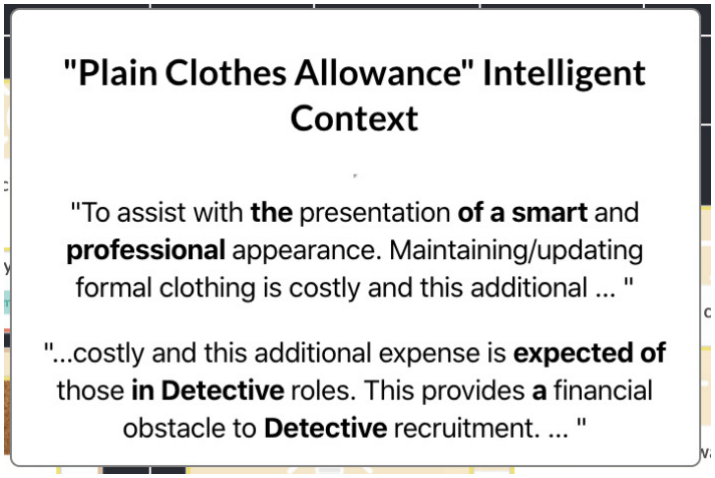

Figure 41 - Idea intelligent context

The motivation behind this is to allow a user to understand what an idea is about without fully committing to viewing the entirety of the content. The context generation looks through the terms in the ideas title, and the terms in the idea's cluster's top themes. It then finds the sentences in that idea around these terms, so the user can understand how the terms are used.

```
// If start of string to index of term is less than 8
if(tokenIndex < 8) {
  const amountToAdd = (surroundingContext - tokenIndex)
  // We add the difference to the post context
  postSurroundingContext += amountToAdd
} else if ((tokens.length - tokenIndex) < surroundingContext) {
  // Or if there's fewer than 8 words after the term
  // Add to start
  const amountToAdd = (surroundingContext - (tokens.length - tokenIndex))
  preSurroundingContext += amountToAdd
}

if((tokenIndex - preSurroundingContext) > 0) termContext += '...'
// If index - surroundingContext, we must use 0 as the start, else a minus number selects from end of list
const slicedTokens = tokens.slice((tokenIndex - preSurroundingContext) < 0 ? 0 : (tokenIndex - preSurroundingContext), tokenIndex +
postSurroundingContext);
slicedTokens.forEach(token => termContext += `${token} `)
if((tokenIndex + postSurroundingContext) < tokens.length) termContext += '...'
return termContext
```

Figure 42 - Intelligent context calculation

An arbitrary number for the number of surrounding words is chosen, in the case of the project, ten. The number of words surrounding the given term is adjusted depending on how close the term is to the start or end of the idea. If a term is at the start of the idea, more context is added following the word, or vice versa if it is at the end.

The user may then leverage the idea similarity doc2vec calculation from the server by clicking on the idea.

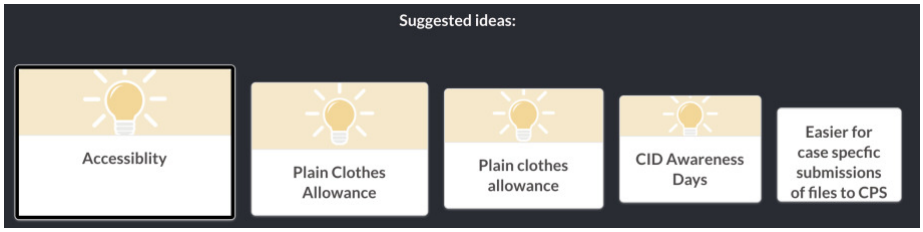

Figure 43 - Suggested ideas

The chosen idea is sent to the server to be analysed using the doc2vec model, which returns an ordered list of the closest document vectors and their associated ideas. These ideas are visualised below the map along with the selected idea. The ideas are ordered from left-to-right in order of similarity to the selected one, and more similar ideas are physically larger. This visualisation was added very late in the development process, following a breakthrough in the usage of doc2vec. This by extension meant that the visualisation of similar ideas is largely unfinished and limited. Figure 43 is a very basic interface for showing similar ideas, demonstrating how this could be potentially done in the future. The author hopes that given more time this could be made more useful and more visually stimulating, but unfortunately time was the limiting factor during the development process.

7.2.4 Topic Model Analysis

The secondary user flow involves investigation of the trends uncovered by topic model analysis. These can be seen in Figure 33, which shows how the topics from the entire dataset and each individual cluster is visualised. The important part of this flow is the method by which the user understands the topics, as the individual words themselves are not hugely useful.

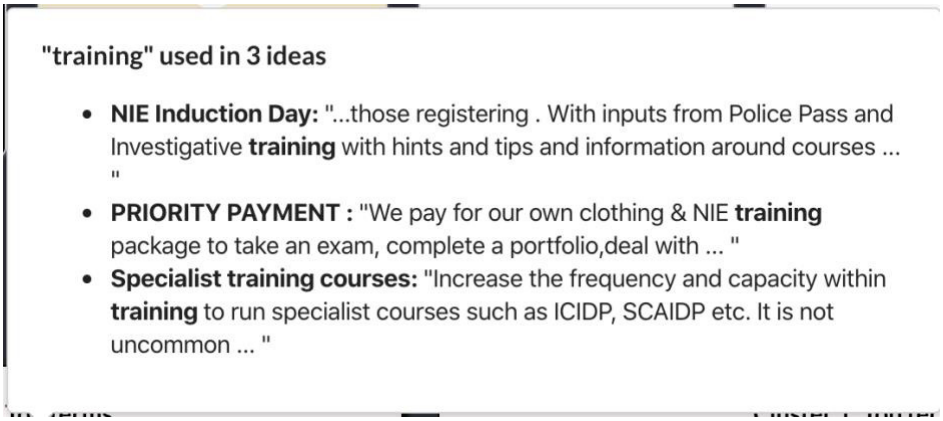

Figure 44 - Topic model context

Each term from the top themes uses a similar intelligent context model to that of each individual idea. Hovering over a term opens this bubble – where each idea is tested to understand whether it contains the chosen term, and if so, runs the same context calculation as shown in Figure 42. Each corresponding idea is named, with a sentence from that idea with the term's usage. If the term is then clicked, it will become bolded, and the idea map will adjust to only show the ideas containing that term. The user can then follow the original idea map flow discussed previously and view all ideas containing that word.

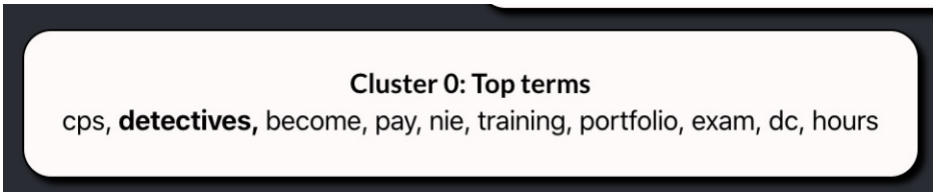

Figure 45 - Selected top term

7.2.5 Limitations

As with any limited development project, issues were encountered along the way. The visualisation was subject to a mixture of limitations with used libraries, and over-ambitious aims. The chosen nivo library proved difficult to use as time went on. The library was not built to support the bespoke visualisation intended – A manipulatable map of ideas that use React components as nodes. Once this approach was decided , it was realised mid development that nivo would require a lot of work to meet this use case. The author was forced to use workaround solutions, such as using the HTML foreignObject component to render a div inside a SVG graphic.

```
<g transform={`translate(${x},${y})`}>
  <foreignObject x="-180" y="-90" width={360} height={240} >
    <div className={`cluster-idea-card idea-${idea._id}`}>
      <IdeaCard idea={idea} node nodeColor={color} viewedIdeas={viewedIdeas} />
      {isChosenIdea ?
        <Button size='huge' as={'a'} href={`https://app.simplydo.co.uk/ideas/${idea._id}`} target="_blank" content='Open idea' />
      : <div style={{height: 50}}/>}
    </div>
  </foreignObject>
</g>
```

Figure 46 - Div inside SVG

This caused a disproportionate amount of time spent focusing on the idea map. If more time was spent researching and testing possible solutions before full development began, this could have been avoided and an alternate solution chosen. As these issues were found mid-development, it was too late to completely pivot.

Another issue was not having strict enough requirements for the visual design. Too much time was spent "playing" with the visualisation, making adjustments until it looked good. This

also became an easy excuse to do simplistic work as opposed to making progress, extending the development lifecycle. Having stricter requirements for visual design and a clearer plan would've reduced this burden. Another alternative would be to have earlier visual design evaluations with stakeholders.

7.2.6 Local API

```
req (method, path, data, success, fail) {
  const xhr = new XMLHttpRequest();
  xhr.open(method, `http://localhost:5000${path}`);
  xhr.setRequestHeader('Content-Type', 'application/json');
  xhr.onreadystatechange = () => {
    if (xhr.readyState === 4) {
      if (xhr.status === 200) {
        let response;
        try { response = JSON.parse(xhr.responseText); }
        catch (err) { }
        if (success) success(response);
      }
      else {
        let message;
        try { message = JSON.parse(xhr.responseText).message; }
        catch (err) { if (fail) fail({ status: xhr.status, message: 'There was a problem with this request' }); }
        if (fail) fail({ status: xhr.status, message });
      }
    }
  };
  xhr.send(data && JSON.stringify(data));
},

request(method, path, data, success, fail) {
  api.req(method, path, data, success, fail);
},
```

Figure 47 - API Code

The API code itself is a very straightforward, standard implementation of the JavaScript XMLHttpRequest functionality. This implementation opens a connection with the given server, sends it data, and waits for a response code to indicate the next steps. The project implementation is also simplistic. Due to time constraints it was decided not to spend time building out an extremely robust, extensible API, but rather to use straightforward requests and only handle basic success or fail responses.

8 Results and Evaluation

Full user testing was heavily constrained due to the company policy of working from home during the national SARS-CoV-2 epidemic. This limited the extent to which the author could conduct full user tests, as staff member resource was stretched when forced to use virtual communications, and observation was difficult. This will be reflected in the methodology.

8.1 Methodology Limitations

Under ideal circumstances, each user would be able to independently operate the program to eliminate any bias and fully understand the steps they would have taken without testing supervision of the author. However, the author encountered the following difficulties which prevented taking this approach to evaluation:

- **Development time:** The time of development of the project extended greater than expected. This was due to over-optimistic planning during development phase coupled with logistical difficulties when adjusting to the work environment during the epidemic. This reduced the capacity to produce a "working" copy of the software, that could be sent to staff members for them to test at home.
- **Skill limitations:** Operating the system requires some degree of experience in computer science. The scope of the project didn't include programming deployment for the project, so upon completion it required installing requirements before launch and command line commands.
- **Environmental factors:** The move to home working meant all meetings moved virtually, limiting the effectiveness of communication, and the project coinciding with other project launches within the company reduced the available time. The author made the decision to use less labour-intensive evaluation methods to offer the staff members flexibility without impacting their existing workflow.

8.2 Final Methodology

The author settled upon a two stage, qualitative evaluation process to understand the effectiveness of the analysis results (Gediga et al. 2002). The nature of the project renders precise quantitative, confirmatory difficult – exploratory data mining uncovers previously unseen data, and by design the project does not aim to produce a specific result, but rather aid an existing process. Its value is based on the end user's opinion of the results, and how the process compares to what they are used to. The author opted to use analysis of historic challenges that meet certain criteria that can be compared to the output from the project. The criteria are as such:

- The challenge was directly assisted by a staff member for at least the management of ideas.
- The author has direct contact with this staff member, and the staff member has capacity in their timetable to spend time with the author conducting the evaluation.
- The challenge has a report with manual analysis, that can be compared with the results of the project.

Two challenges met these criteria, and this evaluation could be completed towards the latter stages of development, dependent on availability to discuss these challenges. This was eventually reduced to one due to time constraints.

Additionally, an earlier evaluation would take place with a wider selection of team members to get a general gauged opinion of the user interface and user experience. The author hopes that these two evaluation periods will help iterative evolution of the project throughout the project process, understand the success of the results, and guide any future work.

8.2.1 Group Evaluation

8.2.1.1 Aim

The first evaluation task, the group evaluation, aimed to determine whether the basic flow of the app suited the needs of the company, as well as an overall appraisal of the visual design. It consulted the team with a casual, semi-structured set of questions and a demonstration to uncover outcomes in a limited period of time.

8.2.1.2 Method

The task was undertaken on the 13th April 2020. The evaluation was completed virtually, where the author demonstrated an example of the standard user flow using ideas from a real challenge in the platform and invited semi-constructed feedback, which the author noted on a table.

In order to complete the session within a one-hour time slot, the session attempted to capture three metrics – What they feel the project succeeds with, where they see shortcomings, and suggestions they had that may improve the experience and future development. The steps below were the ones taken during the evaluation process. This evaluation took place before the doc2vec idea suggestions were implemented, and therefore are not included at this stage.

- The set of ideas were uploaded using the upload button.
- When results received, the author showed the top themes for the clusters, and the context for some of these themes.
- The author demonstrated how the user can use the controls to traverse the idea map, including panning, zooming, and adjusting the number of clusters.
- The author showed how the user could extract knowledge from how ideas are grouped together, and how they may find useful information from outliers.

The table of observations are included below, in a raw format how they were collected. As the data collection was conducted in real time, the sentences consist of quick thoughts and feelings without a great deal of structure. Some of this feedback was delivered by text through a company instant messaging system.

Successes
“Sleek, attractive user interface”
“Ideas as a map is unique and engaging, makes it more “enjoyable””   Additional comments related to this: “The keyboard controls are very cool”, “Animation, visual design of map is also suitably good”
“Design fits with existing systems well”

Multiple comments that the intelligent context is very useful and is one of the most liked elements
“The way clusters found duplicate/very similair ideas is very useful”
“Very “sellable””
“Increasing and decreasing cluster size was cool for metrics”
Shortcomings
Various mentions that some of the trends found weren’t hugely insightful, with broad clusters
“Initially quite intimidating with level of data on display”
Usability problems with overlapping ideas
When ideas don’t have cover images everything blends together (Not sure if this is something that can be solved)
“Having cluster information behind clicks is unintuitive”
Suggestions
“Display overall topics for entire challenge for even more summarisation”
Reduce intensity of map, way of hiding ideas when “dealt with”?”
“Show cluster information readily, without clicks”
“Be able to hide clusters to focus on groups of ideas, good for cooperation”

8.2.1.3 Outcomes

The demonstration was met with a general consensus of positivity and warmth towards the product. Due to the fact the session mainly focused on the method behind the ideation process and not the results, little focus was put on the precision of the analysis data – whether its accuracy could compete with that of a manual analysis. The staff in general were impressed at the themes that had been pulled automatically, and how things were grouped. The idea set contained four ideas that suggested implementation of a “Plain clothes allowance”, and these ideas were clustered and physically grouped correctly, which was useful for demonstration.

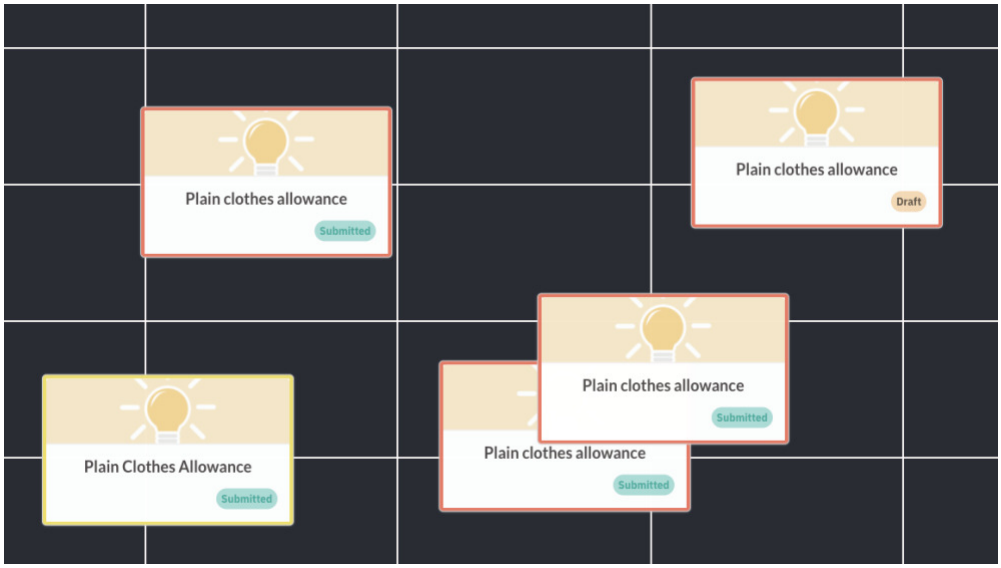

Figure 48 - Closely grouped ideas in idea map

Much of the positive feedback focused on the user interface and visual design. Four of the seven noted comments expressed positivity in how the visualisation looked. The author was confident with the visual design from the outset, as it had been based on previously-justified

style guides developed for the company's products, but confirmation from the team was important – it meant that focus could be moved away from design improvements, purely to functionality. As stated in the table, there were multiple comments about the "intelligent context", where top themes have their context displayed by the usage of the term across the relevant ideas, as seen in Figure 44.

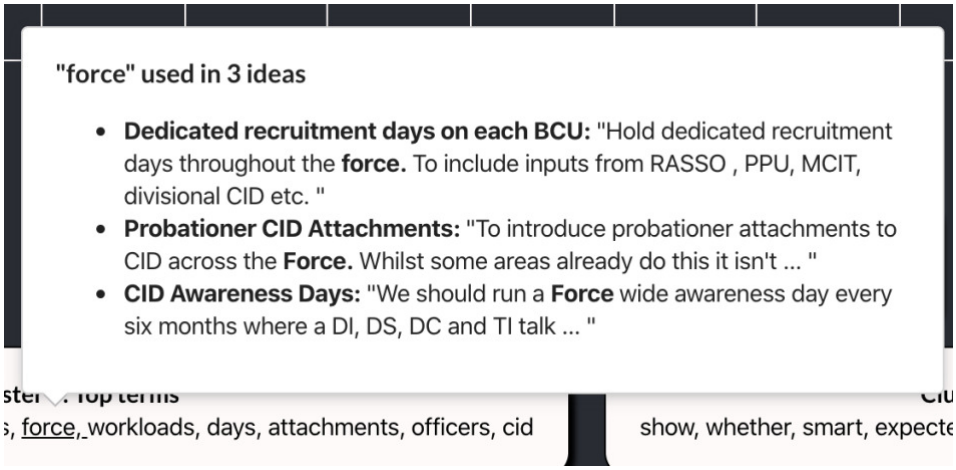

Figure 49 - Intelligent context

As discussed previously, the context system does not do any work on the server side; it is generated when the user hovers over the word, in real time, using the simple function explained in the Implementation section. The positivity surrounding this feature is demonstrative that presentation of information can be as, if not more, important than the analysis itself.

The shortcomings broadly focused on the overwhelming level of information, and the lack of specificity. That some of the trends found are not "useful" is an unfortunate side effect of exploratory data mining, using datasets of this size. This was hypothesised when discussing the theory and implementation – especially topic modelling implementations such as doc2vec, are far less accurate with the size of datasets being used, specifically to the degree where individual clusters are used.

The system was succeeding to uncover clear trends, as shown in Figure 48, but had become increasingly less accurate when there was a great degree of variety among the ideas. Fortunately, this issue has a reduced impact due to inherent bias in the context of challenge based, small scale innovation – Internal research has shown that groups of people naturally focus on specific areas with slightly different viewpoints, which means there are often clear divides between sets of data.

Another main shortcoming was the issue where the idea map would become overwhelmingly cluttered with ideas, especially on first viewing. The point of contention being that while this makes challenges look "well engaged" – successfully gathered lots of ideas – it can also discourage challenge owners due to the intimidating size of the dataset. This spurs future thought on how to reduce the initial cognitive load to ease the user in.

8.2.1.4 Post-Evaluation

Following the evaluation session, the suggestions were taken forward and implemented wherever possible to demonstrate how iterative development can be vital to the success of the product. Changes were made to the visualization appropriately to ensure it was a good as it could feasibly be during the duration of the project.

- Overall top themes for dataset displayed. As seen in Figure 33, the user can see the top terms for the entire dataset rather than just individual clusters, so they can quickly identify what has emerged from the ideation process. Additionally, the top themes for all clusters are always displayed, rather than just when the cluster is selected.
- Slowly reduce the "intensity" of the map, by being able to hide ideas from the map once you have clicked to view them. This can be reversed if needed.
- Provide precise controls for choosing which clusters to display. Hide and show whichever ones the user chooses, giving maximum control.

8.2.2 Comparative Historic Evaluation

8.2.2.1 Aim

The aim of the historic evaluation was to determine a qualitative comparison between manual analysis and the project's analysis. It compares between company reports and the experience of the associated staff member. Results will be a collection of opinions which can be extrapolated.

8.2.2.2 Method

The first comparative exercise was performed with the assistance of the relevant subject matter expert (SME) from Simply Do Ideas, accompanied by the report they produced for the challenge when they managed it and the ideas from the challenge. The challenge was done in tandem with the Future Generations Commission for Wales, with permission to use broad themes and names when discussing.

8.2.2.3 Challenge Background

The author begun by analysing the report created by the SME; the contents of which cannot be included due to data sensitivity and protection but can be broadly discussed. The author picked out interesting things that could be compared with, along with excerpts from the report that could be used to improve the analysis done by the system, and elements that an automatic analysis system would struggle to assist with. These notes were compiled externally with the excerpt below:

- Discusses "power users" – People with highly interacted ideas, that produced multiple ideas or commented often
- In the interest of time spent on a large dataset, the SME pulled out three main themes:
    - "Raising aspirations of young people"
    - "Climate change"
    - "Equiping young people with the skills for the future"
- These discoveries were done via a **time and resource heavy** group meeting in person, due to the large size of the data
- Ideas were "clustered" manually – Points to the observation and research that people naturally group things together, making clustering a natural choice
- Reports on "self-assessed" ideas – The idea template told users to rate the impact, importance and simplicity to implement. The bespoke nature of this is difficult to apply to automatic, non-specific analysis systems like the project's system.

The author noted two clear points of comparison here – The main themes that were pulled out manually, and the fact that all of their analysis required a team of meeting spending time

in person, which comes at a high expense of time and money. Thus, the author decided with the SME two questions that if answered would prove the success of the system in this context:

- Can the system produce clearly similar themes to the ones discovered manually?
- Can the system group ideas to a satisfactory standard without needed a group of people?

It will also be discussed whether these advantages mean the analysis can replace a group of people in areas it cannot perform in – and how this gap can be bridged. In this challenge, this would mean the self-assessment task that is discussed in the manual report.

8.2.2.4 Using the system

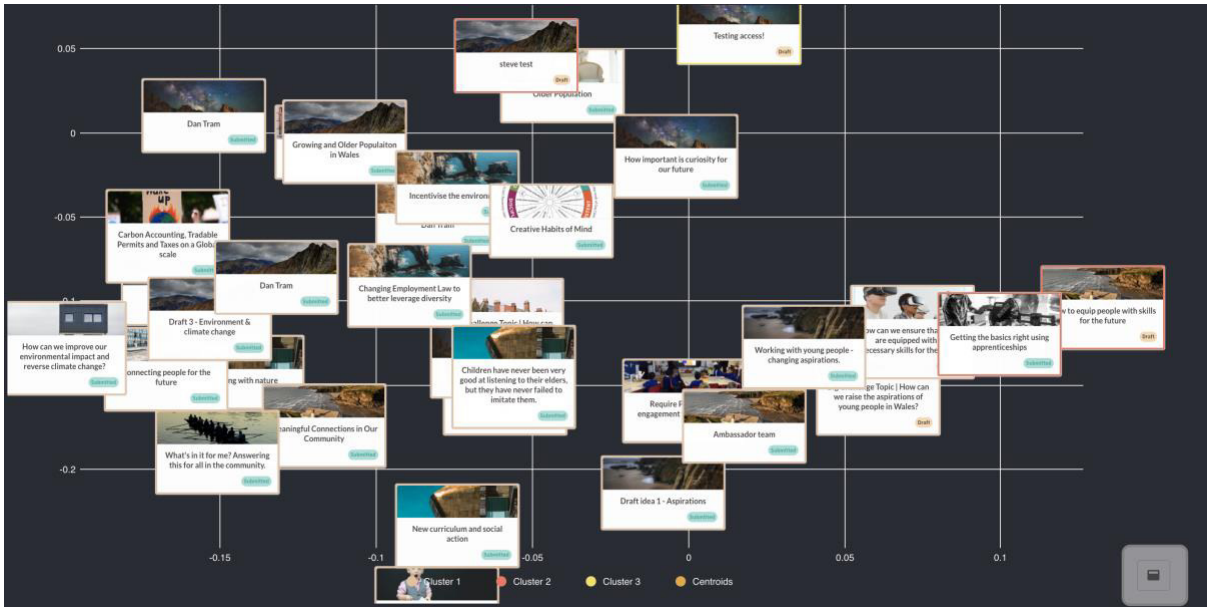

Figure 50 - Idea map for FGCW challenge

In a similar fashion to the group evaluation, the set of ideas were uploaded to the system and the same analytical steps were performed to emulate the predicted approach of the average challenge owner. The author showed the steps taken, and the intermediary results, to the SME and asked him for his thoughts compared to the manual process, and to answer the questions set, to understand whether the project is a success.

The process began by looking at what the topic model analysis had produced for the clusters. As the manual report uncovered three distinct themes, the number of clusters was set to three, to understand whether the analysis could produce similar results. The results below are the top ten themes from the three clusters.

- **Cluster 1:** preventative, need, young, determinants, people, improve, inequalities, food, wales, health
- **Cluster 2:** needs, transport, waste, change, climate, public, need, communities, people, wales
- **Cluster 3:** work, video, wales, ensure, equipped, necessary, future, young, skills, people

Both the author and the SME agreed that the themes uncovered by the system were remarkably similar to those discovered manually. The three manual themes – young people, climate change, and equipping skills, have been distinctly represented within the three clusters here. You can see in Cluster 1 that "young" and "people" are prominent, "change" and "climate" are seen in Cluster 2, and Cluster 3 has "equipped" and "skills". This is a strong demonstration for the success of the analysis, and proves the effectivity of the approach taken. The SME also agreed that these results answer the question "Can the system produce clearly similar themes to the ones discovered manually?" positively, succeeding in this.

Following the demonstration, the author asked the SME for general comments on the process. These thoughts were captured using a notes application and repeated here, including additional thoughts I had during this.

- "Very impressed with acquisition of almost exactly the same topics"
- "In a real setting would reduce time heavily"
- "Still a few analytically useless words such as 'needs'" Focus on nouns only?
- "Overall top terms not actually that useful"
- "Having the Previously Viewed ideas streamlined process"
- "Navigation somewhat cumbersome with this size of data, some overlapping caused difficulty" Perhaps needs smoothening and UI improvements of the map
- 3 clusters seemed to be the preferred number for level of detail while still being fast to sort, especially with dataset of this size
- Noted that this would be really useful AT SCALE where it becomes unfeasible to spend man hours on these reports - but manual still may be more precise and useful sometimes

8.2.2.5 Outcomes

Many of the observed comments mirror those discussed in the group meeting, with repeats of the difficulties encountered with having a large set of ideas on the map. They commented that the addition of hiding viewed ideas, however, improved this. The important part of his comments is deciding whether the product answers the second question of success, "Can the system group ideas to a satisfactory standard without needed a group of people?". The answer is slightly more complex than a simple "Yes" here.

The SME states in his comments that he thinks the project's usefulness is proportional to the scale of the challenge. This makes sense in both the accuracy of the analysis and the amount of time it saves for the user. A smaller challenge with fewer ideas may require less time to sift through the ideas, and consequently less time uncovering the themes through the dataset.

Additionally, the previously discussed research tells us that a smaller number of ideas would lead to less "accurate", and by extension less useful, data mining. Whereas a challenge with many dozens of ideas would require more man power and time to traverse all the ideas, and having more content provides the opportunity for a richer analysis. The other issue encountered is when the challenge owner may be looking for very context specific metrics – for example, the self-assessment scores in this challenge.

Overall, the SME agreed that even during a smaller challenge, the system successfully **aids** the process. Where the shortcomings appear due to sample size and lack of specificity, the system does not impede the regular process, and the user can still take advantage of certain

features where necessary, even if that doesn't mean using the entirety of the rich analysis on offer. The flexibility will be key for future expansion.

8.2.3 Self-Evaluation and Requirements

It is important to understand the author's considerations of how they perceive the success of the project too. Looking at how the final iteration of the project matches the initial requirements is a strong basis, from which it can be discussed where it differs and why.

Green = Full success
Yellow = Semi-success or adjusted through iteration
Red = No success, or removed/changed entirely

Requirement	Did it succeed?
Key Supplementary Tasks	
Data Clean-Up (MINIMUM): System must be able to accept and normalise a large variety of ideas in different formats, to build a standardised corpus of ideas for further comparison, amalgamating the content for text processing.	System has a clean-up file that creates normalises ideas. Performs key tasks including tokenization, stemming and combining content into a single string.
Analysis	
Supports Multiple Use Cases (MINIMUM): The system will support a variety of different skill-levels, supporting a multitude of use-cases. The system will support the user regardless of their experience.	The final product has a user flow designed to cater for a multitude of user experience level's, wants and needs. This has been demonstrated and verified.
Word Frequency (MINIMUM): Word and bigram frequency analysis; allowing simple visualisation of the most used terms. Both segmentation and tokenization will be used, along with a common and user-extended library of stop-words to remove analytically useless connectives, verbs and common terms that have no use for investigation	System calculates word frequency as a biproduct of calculating TF-IDF values, but this is not visualised. The simplicity of the knowledge found was not considered more useful than using data mining methods for discovering popular terms, which eventually replaced this.
TF-IDF (MINIMUM): Analytic techniques to pick out the most 'popular' terms in the set of ideas, which in turn allows discovery of the overall 'key topics'. Acts as basis for clustering and enables broad stroke inspection of challenge outcomes.	TF-IDF values are calculated for every idea in a given set, reduced to a two dimensional representation using PCA, then the coordinates are sent to the visualisation.
Clustering (MINIMUM): Act of separating the ideas into previously unknown, potentially non-discrete groups based on their individual TF-IDF matrices, traits, and the occurrent of trends within the ideas; the aim being to create subsets of ideas that belong to a particular topic within	System leverages scikit k-means to perform cluster data mining on the ideas, and assign every idea a cluster to be visualised. Cluster centroids have their dimensionality reduced to be given coordinates.

the idea corpus, which were previously found via inverse document frequency analysis. This is the end goal to solve the post ideation process of filtering ideas, as the administrators can immediately extract and dissect the key outcomes of the challenge.	
Word Embeddings and Idea Closeness (DESIRABLE): Representing individual words as vectors in a vector space, finding words that have the same representation and thus being able to define some degree of similarity, or closeness in the vector space, between different ideas. Using a pre-existing populated embeddings model for word comparison, we can compare the features of ideas and be able to produce a traversable map of ideas.	Topic modelling and sentence to vector analysis is done on all ideas, in a variety of ways, but the traversable map of ideas was excluded due to limitations in development time. Instead, the results from this data mining was used elsewhere in a less developmentally intensive manner.
Visualise	
Key topics (MINIMUM): Simple visualisation of the key topics acquired from TF-IDF and term frequency analysis data, to enable a minimum viable product	TF-IDF scores are visualised by using a 2D representation of their relationship on a scatter graph. Similar ideas are grouped closely together on the map, while outliers are further away.
Clustering representation (MINIMUM): A well-justified method to represent the clustering of ideas in the dataspace, that fully supports a fast and efficient idea sifting process for the user. Clearly define the extracted key topics and allow granular access to ideas clustered within these topics for deeper analysis.	Ideas on the scatter graph are coloured based on their cluster. Key themes are modelled for each cluster, with appropriate context, and the user has controls to limit which clusters they see. Evaluation verifies that the clusters are identifiable.
Embedding graph (DESIRABLE): Take advantage of the embedding vector-space and similarity data to create a linked map visualisation of the closeness of ideas. Can pick out trends between clusters, navigate the data space, view potential outliers and more.	Embedding graph was not created due to limitations in time for development, and changes in focus following evaluation. The time to develop outweighed the benefit to the user flow, and the idea was removed for now.
Data manipulation (DESIRABLE): Create set of tools that give the user the ability to manipulate the tools that analyse the dataset in real time to allow them to decide their own granularity of analysis – By empowering them to complete actions such as adjusting cluster size, with the	The user can adjust the number of clusters, hide viewed ideas, upload their own ideas, zoom, pan, and choose to view context for different topics.

visualisation robust enough to support these user activities.	

Of the twelve requirements devised on the inception of the project, nine were considered fully successful, two were adjusted to better suit the direction of the project, and one was removed. Any adjustments were made to focus on the creation of a complete product with a full user flow, that successfully aided a user's management of ideas. Both word frequency and document vectors were adjusted to be incorporated in a simpler user flow.

The author felt that using these to create entirely new sections of the analysis and visualisation systems would've exponentially increased both development time and the time needed to evaluate. To still meet the requirements, these features were moved elsewhere to make use of their richness without having an overwhelming user interface. Word frequency is still used as part of the calculations for TF-IDF, and both topic modelling implementations were used for suggesting similar ideas and finding unseen themes from groups of ideas.

The only requirement that was left completely unsatisfied was creating a graph/map to display the data generated from the vector representations of the ideas. This would've been a strong comparison with the graph/map created by using TF-IDF representation and allow the user to look at multiple representations. It was decided during development that the effort required to complete this outweighed its usefulness in the short term, as it created an additional user flow that would need to be bug fixed and evaluated. It is hoped that in the future this could be included.

As development of the project progressed and evaluation took place, it was found that the topic model analysis was more impactful than first thought. While the clustering was considered useful, having the analysis automatically pull out themes proved to be the biggest improvement in time and richness compared to manual analysis. The project evolved to put a bigger focus on this, by providing topic model analysis for the entire dataset, individual clusters and idea suggestions. While this meant a deviation from the original title, the author feels these changes ultimately helped towards the goal of improving the post-ideation process.

The overall progress made towards the requirements, and the outcomes from the critical evaluation with members of the company, the author believes proves that the project is a success. This can be affirmed through the results from evaluation, and progress made towards the requirements.

8.2.4 Additional Evaluation

Following the previous evaluations, the author demonstrated the final project to two members of the Cardiff University Crime and Security Research Institute. While this evaluation wasn't a formal procedure, it helped guide future work and aided the author's understanding of the user story – running through the project with the same mindset as a challenge owner. The application was presented by screen share presentation over an online video call and was followed by an informal discussion of the members' thoughts on how the project succeeded.

8.2.5 Evaluation Conclusions

The project set out to attempt to aid the bottleneck of manual analysis in the ideation process, so it is important to consider whether the evaluation proves that this aim has been achieved. This can be primarily extrapolated from the outcomes of the historic evaluation – this is where the discussion of the effectiveness, speed and usefulness of the analysis was directly addressed. The SME crucially stated their satisfaction that the results from the project mirrored those found manually, using a team of people, office space and valuable work hours.

To this end, we can conclude that in this instance, the project produced **very similar** results, in a **dramatically shorter** time, using **far fewer** resources – the data mining found the same themes as a group of people. If the project would've been used for this challenge, the need to gather people together using both valuable time and money would've been reduced – successfully solving the analysis bottleneck. In the future a greater number of evaluation exercises would need to take place to confirm this fully, but within the constraints of the project, it would be fair to say that significant progress has been made towards solving this problem.

8.2.6 Methodology Appraisal

The process behind development was mostly a success. The technologies used mirrored those used in Simply Do Ideas application stack, and due to the familiarity of the author no difficulties were encountered using those tools.

The development of the project used a combination of waterfall and agile development. It began with a thorough stage of research and planning to devise the aims of the project and plan the basics of how it will act and look. Following this stage, development progress would be relayed to relevant stakeholders in small bursts with screenshots, and any changes made as necessary. This approach ensured the project maintained its focus, while changes were made on a rolling basis before it got too late.

It may have been preferable to do an iterative review following the planning process – This would've meant any changes could've been made before the development process began. This method also could've meant that potential difficulties – such as the removed embedding graph – were discussed before development began and could've been adjusted in the initial requirements to make them more achievable.

The author also believes that beginning writing the report earlier in the development process may have aided some of the decision making. Report writing is a natural extension to the "rubber duck" development technique, and they uncovered issues and found ideas through writing these decisions in the report.

9 Future Work

Hofstadter's Law says that "Everything takes longer than you think, even when you take into account Hofstadter's Law" (Kreuzer and Robbiano 2005), and this project is no different. Simply Do Ideas is also a SaaS (Software as a Service) company, so their software is constantly growing and evolving – it is not contained to a single time period. The duration of the project only allowed a limited period of development, but the author discovered a number of ways the project could be taken further in the future.

- **Larger datasets:** As discovered from evaluation and research, the dataset size used for the topic modelling is considered "toy size" – with a less-than-great accuracy of results. Future work should consider research into how exactly the results of the analysis is affected by the number of ideas. This may guide whether to use the analysis system for sifting ideas, or to do it manually.
- **Further doc2vec:** Reflection on results and requirements show that the usage of document vectors were pushed down on the requirements as they proved to be less relevant to creating a strong minimum value product. Future work should further this work, given the foundations are in place including a full doc2vec modelling process. Using the advantage of larger datasets, the original requirement to create a "map" using document vectors could be completed, and compared to the "map" created from tf-idf.
- **Implement into existing product stack:** One of the clearest goals for the project is for it to be implemented into the existing Simply Do Ideas platform, effectively commercialising the system. This would allow it to continue directly when a challenge is completed, without having to download then upload ideas. This would require further iteration until the project is seemed "ready for sale", where it can begin to be used by real world clients.
- **UX Simplification:** Multiple comments during the evaluation stated how the initial idea map and set of topics were daunting when first seen. This is due to the large amount of data on the screen. Reducing this cognitive load could be key to increasing usability and the number of users who trust and leverage the capabilities of the analysis. Ideally the developer taking it further could use an experience user experience designer to justify methods to achieve this.
- **Expanded suggestions:** The suggestions proved to be a popular feature of analysis, and expansion of this could prove fruitful. An unrealised idea that the author had during development was to increase the scope of suggestions to include external resources. The doc2vec model could include webpages, online journals, scientific research and more. Ideas could then suggest similar content from an extremely vast library of information. For example, an idea on improving the office could suggest furniture shopping results, research papers on optimal office layout, or exemplary images of offices. The "intelligence" of the analysis is key to how "sellable" it is.
- **Topic summarisation:** A future feature discussed with the members from the Crime and Security Institute was summarising the top themes from the corpus and from each cluster. One possible method for this would be utilising WordNet[13], a large database of words and synonyms to find the closest synonym to the group of words and present the topics with a single word or phrase.

[13] https://wordnet.princeton.edu/

10 Conclusions

The project set out to offer a solution to the issue of handling a large set of complex data. Its application could spread beyond merely the idea context, but the scope of the project was to demonstrate how capable data mining and strong visualisation can be in the management of ideas. The project had to target a number of use cases, with a variety of target users that have different degrees of skill level. The project also included developing both a server and a visualisation, with justification and evaluation for the effectiveness of both, using bespoke data adhering to existing user interface guidelines. All these considerations led to a complex set of aims and requirements, with a carefully considered scope.

By the conclusion of the project, the author successfully produced a clear minimum viable product for both the analysis server and the visualisation front-end.

The analysis server is able to reason and normalise a variety of ideas, calculate tf-idf for the entirety of the corpus and use this to k-means cluster the ideas, then use exploratory topic modelling to uncover the themes among the ideas without any manual assistance. Following this, it could use doc2vec to find similarities between ideas and suggest them when requested. This analysis would be packaged in a reasonable format and returned to the client that made the request.

To make this analysis human-understandable, a robust visualisation was developed that used interesting, engaging techniques to display the data. An idea “map” displays a two-dimensional representation of the relationships between the tf-idf values for each idea, enabling the user to recognise the variation in the dataset, visualising similarities and differences in a physical space.

This idea map could be manipulated through zooming and panning for maximum control. The k-means clusters would be represented via coloured borders on the ideas, allowing the user to quickly see where groups of trends formed. These clusters could be investigated to see the trends that were data mined within them, and the context surrounding these trends. This allowed the user to instantly understand what trends exist in the dataset, as well as frames of reference for these trends, without having to open a single idea.

If a user chooses to investigate a single idea to view its full content, the visualisation will leverage the doc2vec data to suggest similar ideas that they can investigate next for further information about emerging trends. The visualisation also offered the user optional controls to manipulate the granularity of the analysis – by choosing their own k, the number of clusters used during k-means.

The effectiveness of these techniques were verified through numerous stages of iteration and evaluation. Subject matter experts were consulted and offered a demonstration of the analysis system in action. The author compared how the analysis performed through data mining, to the manual analysis that was performed when the historical challenge was completed.

Evidence displayed that the data mined themes were almost entirely similar to those found manually, which required a physical group meeting using multiple people. The SME agreed that the project would be of definite assistance to the ideation process when the number of ideas is far greater than manageable by a single person. The design of the user interface was also

evaluated by members of the Simply Do Ideas team, and iterated upon to maximise the effectivity, usability and visual design.

Ultimately, the system achieved the goals it set out to perform. Some of the initial concepts deviated – the importance of clustering was found to be less than expected, while the usefulness of topic modelling greater than expected. Some features were changed or removed, such as the doc2vec embeddings map. But the final product developed was proven to meet the broad aim of streamlining and improving the management of ideas. The author has reflected on how the system could further enhance the experience with more time, but the existing system is a clear indicator of the effectivity of data mining, language processing and engaging visualisation in this context, and perhaps the wider context of managing knowledgebases.

11 Reflection on Learning

While the author had a wealth of experience in front-end development, the venture into exploratory data analysis was one of little knowledge. The aim when devising the goals was to explore the familiar context of ideation but doing so while investigating the capability of data mining. Their university degree had little focus on front-end development, which was their full-time job, while they had taken numerous modules focused on more traditional programming, as well as the management of knowledge. Their interest in the idea of knowledge management led to taking the approach of trying to investigate interesting techniques for doing so.

The interest in discovering techniques for manipulating data was key to motivation during the duration of the project. Being able to use as many as possible, remaining ambitious in the scope, proved fruitful – the author was able to include a number of data mining techniques from various subject areas. The unfortunate side effect of this was that the greater amount of data that was created, the more visualisation components were needed. The time taken to visualise data took a disproportionate amount of time compared to the time needed to create the analysis tools.

While Python has a wealth of extremely versatile and easy-to-use data mining libraries, there were fewer bespoke solutions for React, especially for the approach taken to display ideas in a map. The aims set out meant that the quality of the analysis data was only as good as how it was presented. The author found that this was impactful on the motivation to work. The author was working as a front-end developer alongside the duration of the project, so the drive to do more front-end development on the project was diminished. In hindsight, the project would've more interesting had it solely focused on producing analytical data. Even more ambitious tools could've been used with a great focus on confirmatory hypothesis and comparing the effectiveness.

This also demonstrated how a move to a "test-first" development approach could've benefited progress. The procedure taken was to complete development first, **then** decide how the product would be evaluated.

The author then encountered severe difficulty in finding methods to scientifically evaluate their progress. The heavy focus on visualisation made the usage of quantitative very difficult, and the end result was a reliance on qualitative evaluation with staff members. While the outcome from this evaluation was surprisingly positive and effective, a preferable approach may have been to devise quantitative tests that could verify and compare how the analysis performs, which the ability to compare the effectiveness of different data mining techniques, including numerical figures such as accuracy. This also would've been aided by a greater focus on the backend rather than visualisation.

A conscious decision was made to research why techniques are used, as well as how they were used. This meant reading academic papers and guides on the usage of different mining techniques in different contexts. The double-loop learning approach of questioning your assumptions behind the actions you take (Argyris 2002) was one the author took on board during their Knowledge Management module, and proved a dramatic change to their standard work flow. The understanding of why you are doing something is a great aid to scientific report writing – the advantage of being able to confidently justify why you take your actions. The

author plans to apply this technique in their future learning and professional development; it has proved to be instrumental in being confident in your decisions.

The author was overall happy with the approach to the project. The scope required little adjustment; barring some “crunch” development, all the goals were suitably met. The timeline devised in the initial report ended up rarely adhered too – the author was accustomed to agile development from their job and found that it was far more effective to decide the order of development in shorter bursts rather than an overarching, waterfall timeline. The project also taught to not be against taking a scholarly approach to tasks. The act of writing a report, having to verify all decisions made and extensively evaluate them, ensured that development always met a certain quality.

12 Table of Abbreviations

TF-IDF	Term Frequency – Inverse Document Frequency
SME	Subject Matter Expert
LDA	Latent Dirichlet Allocation
NLP	Natural Language Processing
UI	User Interface
HTTP	HyperText Transfer Protocol
JS	JavaScript

References

Anderson, D. J. 2010. Kanban: successful evolutionary change for your technology business. Blue Hole Press.

Argyris, C. 2002. Double-loop learning, teaching, and research. Academy of management learning & education 1(2), pp. 206-218.

Benitte , R. 2019. ScatterPlot | nivo.

Branan, N. 2010. Are our brains wired for categorization. Scientific American,

Brownlee, J. 2019. Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates. http://machinelearningmastery.com.

Budhiraja, A. 2018. A simple explanation of document embeddings generated using Doc2Vec. Medium.

Cameron, D. 2013. A Software Engineer Learns HTML5, JavaScript and jQuery. CreateSpace Independent Publishing Platform.

Chire. 2011. K-means Gaussian Data.

Duan, L. et al. 2009. Cluster-based outlier detection. Annals of Operations Research 168(1), pp. 151-168.

Gama, P. 2007. Stop reinventing the wheel. IEEE Distributed Systems Online (5), p. 9.

Gediga, G. et al. 2002. Evaluation of software systems. Encyclopedia of computer science and technology 45(supplement 30), pp. 127-153.

Goldberg, Y. and Levy, O. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722,

Grinberg, M. 2018. Flask web development: developing web applications with python. " O'Reilly Media, Inc.".

Heck, R. H. 1998. Factor analysis: Exploratory and confirmatory approaches. Modern methods for business research, pp. 177-215.

Iglesias, M. 2019. Using Your Library with React.Pro D3. js. Springer, pp. 201-218.

Kanungo, T. et al. eds. 2000. The analysis of a simple k-means clustering algorithm. Proceedings of the sixteenth annual symposium on Computational geometry.

Koç, S. Ş. et al. 2018. Triadic co-clustering of users, issues and sentiments in political tweets. Expert Systems with Applications 100, pp. 79-94.

Kreuzer, M. and Robbiano, L. 2005. Computational commutative algebra 2. Springer Science & Business Media.

Lau, J. H. and Baldwin, T. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. arXiv preprint arXiv:1607.05368,

Le, Q. and Mikolov, T. eds. 2014. Distributed representations of sentences and documents. International conference on machine learning.

Lloyd, S. 1982. Least squares quantization in PCM. IEEE transactions on information theory 28(2), pp. 129-137.

Loper, E. and Bird, S. 2002. NLTK: the natural language toolkit. arXiv preprint cs/0205028,

MacQueen, J. ed. 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Oakland, CA, USA.

Nandi, R. N. et al. eds. 2018. Bangla news recommendation using doc2vec. 2018 International Conference on Bangla Speech and Language Processing (ICBSLP). IEEE.

Newman, I. et al. 1998. Qualitative-quantitative research methodology: Exploring the interactive continuum. SIU Press.

Organization, W. H. 2020. Coronavirus disease 2019 (COVID-19): situation report, 72.

Pedregosa, F. et al. 2011. Scikit-learn: Machine learning in Python. the Journal of machine Learning research 12, pp. 2825-2830.

Porter, M. F. 2001. Snowball: A language for stemming algorithms.

Ramos, J. ed. 2003. Using tf-idf to determine word relevance in document queries. Proceedings of the first instructional conference on machine learning. Piscataway, NJ.

Rehurek, R. and Sojka, P. eds. 2010. Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Citeseer.

Rehurek, R. and Sojka, P. 2011. Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic 3(2),

Richardson, L. 2007. Beautiful soup documentation. April,

Robbestad, S. A. 2016. ReactJS blueprints. Packt Publishing Ltd.

Sadighpour , S. 2016. arxiv-doc2vec-recommender.

Siddiqui, K. 2013. Heuristics for sample size determination in multivariate statistical techniques. World Applied Sciences Journal 27(2), pp. 285-287.

VanderPlas, J. 2016. Python data science handbook: Essential tools for working with data. " O'Reilly Media, Inc.".

Wold, S. et al. 1987. Principal component analysis. Chemometrics and intelligent laboratory systems 2(1-3), pp. 37-52.

Yang, S. et al. eds. 2019. Metamorphic exploration of an unsupervised clustering program. 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET). IEEE.

Zhu, N. Q. 2013. Data visualization with D3. js cookbook. Packt Publishing Ltd.

13 Supplementary Material

13.1 Stop words

'would', 'use', 'using', 'year', 'could', 'useful', 'use', 'idea', 'used', 'save', 'know', 'knows', 'get', 'us', 'etc', 'make', 'able', 'add', 'easy', 'making', 'https', 'com', 'aspirations', 'embed', 'youtube', 'vimeo', 'couk', 'www', 'http'