D1 Report

Matthew Frankland Farhan Hafiz George Hughes

Sabina Jędrzejczyk Ross Moir Ridwan Mukhtar

Mark Schmieg Nicolas Sparagano Cailean Welsh

September 2020

Contents

List of Figures

List of Tables

1 Introduction

Topic Maps is a tool developed by the Strategic Futures Laboratory at Heriot-Watt University. The tool has a pipeline and an interface to interactively view information about topics. The topics, when clicked on, show different papers associated with the topic and how relevant they are. It uses the Latent Dirichlet Allocation (LDA) algorithm for its topic modeling and uses this to extract topics from the supplied texts.

1.1 Motivation

The motivation behind this project was due to the fact that the Latent Dirichlet Allocation algorithm is stochastic in nature. Models can differ from each other, not only due to different parameters being used in their creation, but on the seed provided to the random number generator upon initialisation. As seen in Figure 1, each iteration produced different solutions. Each solution can be valid even though each solution can have different parameters.

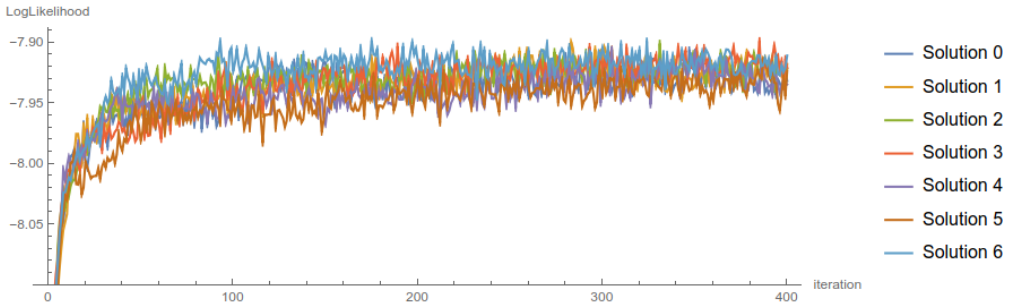

Figure 1: Log-Likelihood

As seen above, it is essential that there is a tool that can investigate these models based on various metrics. This tool will help give users the analysis needed to decide which model is best suited for the task and prove that the chosen model is valid based on a variety of calculated metrics.

1.2 Goals and Objectives

The goal for this project is to develop a tool that uses various metrics to investigate models that are produced. This allows for users to see which model best suits their needs and can also be used to convince other people why the chosen model is appropriate and reliable. The goals for this project are:

Goal 1	Research different ways to validate the generated models
Goal 2	Develop a tool to run multiple instances of mallet over a range of parameters
Goal 3	Calculate the perceptual and quality differences of the models
Goal 4	Visualise the comparison of the models

Goal 1 is essential due to the many different models that are being produced. Researching different metrics is important to help validate which model is best suited for its task.

Goal 2 is important as seen in the figure above; different parameters and iterations produce different solutions and as a result, it is important that the tool can take in multiple models.

Goal 3 takes the research undertaken in goal 1 and implements this to analyse the models. This gives users analytical information about the models such as the perplexity and coherence of a given model.

Goal 4 allows users to easily see the differences between the models in a visual manner, such as in a graph. This enables users to easily ascertain the models that best suits their needs.

2 Background

2.1 Topic Models

Topic modeling is an unsupervised machine learning method which is used to extract topics from given bodies of text. It works by giving weightings to a selected set of words, also referred to as tokens. Topic models are stochastic in nature and as a result, can produce various different models. Topic modeling is a useful way of discovering the semantics of a given document[7].

2.1.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a popular form of topic modelling [8]. In LDA, documents are random mixtures of latent topics where each topic is represented by the distribution over the words [9]. LDA performs the following process:

For each word in a given document:

1. Pick a topic based on the multinomial distribution of a document
2. Pick a word from the multinomial distribution of a topic [10]

2.1.2 Gibbs Sampling

Gibbs Sampling is a Monte Carlo Markov-chain algorithm. It is a sampling algorithm used in topic modelling and is one of the most commonly used within this domain.

This algorithm works by getting a topic for each word in a document. This is usually done by calculating the normalization constant and getting the topic based on the probability that is calculated. This is repeated for all the words in the document [11].

The Gibbs Sampling formula (which is not in the scope of this project) is comprised of two parts. Part one is about how much a topic is represented in a document. Part two is about how many times a word from a document was assigned to a given topic.

The following will show a step by step example how the word "world" is assigned to a topic using the Gibbs Sampling technique.

India	**enters**	**world**	**cup**	**final**
1	3	1	2	4

Table 1: Topics assigned to words [1]

Table 1 shows that the word "world" was assigned to topic 1, but this example will verify that "world" is correctly assigned to topic 1.

	Topic 1	**Topic 2**	**Topic 3**	**Topic 4**
India	70	5	0	8
enters	2	3	15	6
world	28	4	12	1
cup	6	43	6	0
final	7	0	9	31

Table 2: Number of words per topic [1]

In Table 2, above, it can be seen that the different words in the document have been assigned to the different topics. One can see that "world" has been assigned 28 times to topic 1. To verify the correctness of this assignment for this word instance, 1 will be subtracted from the total number of assignments of "world" to topic 1. This leaves the total at 27. Figure 2 shows how much a topic is represented in the document (Part 1).

Figure 2: The presence of a topic across documents [1]

The different bars seen in the figure above, display how many times the different topics are represented in the document that is being processed. The example in Figure 3 shows the number of times the word "world" appears in the different topics (Part 2).

Figure 3: The presence of "world" across topics [1]

In the Figure 3, above, the bars represent the amount of times the word "world" has been assigned to the different topics. The bars seen in the image are the graphical representation of "world" from Figure 2.

[1]

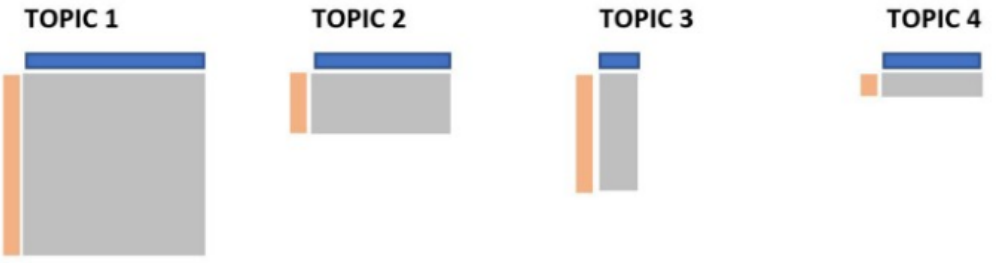

Figure 4: The combination of Figure 2 and Figure 3

This figure shows the result of combining the values from Part 1 (Figure 2) and Part 2 (Figure 3). In a normal execution of Gibbs Sampling "world" would be probabilistically assigned to a topic. However, in topic modelling, to make a topic more pertinent, one would first do a normal execution of Gibbs Sampling, then run Gibbs Sampling again where words would be assigned to the topic with the highest probability. In this case, if the probability determines what topic is assigned to what word, then "world" would be assigned to topic 1. For a whole document, this process will be reproduced multiple times to achieve the best topic combination possible.

2.2 MALLET

MALLET is a statistical natural language processing tool that includes an implementation of LDA Topic Modelling. MALLET uses a scalable version of Gibbs Sampling, and includes tools for inferring topics for new documents given trained models [12].

2.2.1 MALLET Parameters

MALLET has several parameters that can be used to help build and optimise its generated models. The following tables describe some of these parameters ([13],[14]).

Building Topic Models These parameters define the number of iterations and topics used when creating a model.

Parameter	Description
`--input [FILE]`	This is a file that has been converted into the .mallet format.
`--num-topics [NUMBER]`	This is used to determine how many topics for MALLET to choose. For example, setting this to 20 will cause 20 topics to be chosen.
`--num-iterations [NUMBER]`	This is the number of sampling iterations. The higher this number is, the longer it will take to complete the models. However, the quality of these models will increase.

Hyperparameter Optimisation This is used to optimise the hyperparameters of the models. This allows the model to fit the data better. The parameters are used to turn on this feature and to allow for optimisation to occur over N iterations.

Parameter	Description
`--optimize-interval [NUMBER]`	This turns on hyperparameter optimisation.
`--optimize-burn-in [NUMBER]`	This determines the number of iterations that need to take place before hyperparameter optimisation is ran.

Model Output These parameters deal with the files that store different outputs that MALLET offers.

Parameter	Description
`--output-model [FILENAME]`	This is the name of the file within which to store the serialised topic trainer.
`--output-state [FILENAME]`	This stores the words in a piece of text with its associated corresponding topics.
`--output-doc-topics [FILENAME]`	This is the name of the file that stores the topics associated with the documents.
`--output-topic-keys [FILENAME]`	This stores the top words and the Dirichlet score for each topic.

Topic Inference This contains a file which contains the parameters used for finding topics in a different document.

Parameter	Description
`--inferencer-filename [FILENAME]`	This creates a topic inference tool based on the generated model.

Topic Held-Out Probability This is used to estimate the log probability over all the topics.

Parameter	Description
`--evaluator-filename [FILENAME]`	This is used to obtain the information based on the held-out probability estimate.

Diagnostic File Another useful parameter is: *–diagnostics-file [FILENAME].* This parameter is used to obtain the diagnostics of a model which can provide useful information. Some of the values that are stored/calculated are:

Diagnostic Value	Description
`Tokens`	This is the number of words associated with a topic.
`Document entropy`	This calculates the probability of the documents for a topic
`Coherence`	This calculates the probability of the words in a topic occurring together.
`Exclusivity`	This calculates how many words within a topic are exclusive when compared to other topics.

2.2.2 MALLET Output

The training data takes the following format:

1. The program starts by breaking down the number and size of tokens, topics, max tokens and total number of tokens set within the program and the input.

2. The program then gives the log likelihood divided by the number of tokens for each iteration of the topic model.

3. The program then prints out the key words, the words that help define a statistically significant topic, per the routine.

An example of this can be seen in Figure 5, below, where the first column indicates topic ID, the second column is the log-likelihood and the third column is the key words associated with the topic.

training.txt

```
1   Mallet LDA: 100 topics, 7 topic bits, 1111111 topic mask
2   max tokens: 655
3   total tokens: 457721
4   <10> LL/token: -10.38923
5   <20> LL/token: -9.92184
6   <30> LL/token: -9.71719
7   <40> LL/token: -9.59206
8
9   0   0.01    year band record time years hair include
10  1   0.01    troops israel army iran military soldiers official
11  2   0.01    house reagan president general bush bill office
12  3   0.01    market futures prices demand trading day don't
13  4   0.01    billion budget bush million year spending congress
14  5   0.01    state abortion tuesday years reed building officials
15  6   0.01    art mother museum show york home house
```

Figure 5: MALLET Training Data

Another example of output data can be seen in Figure 6. The program outputs every word in your corpus of materials and the topic it belongs to.

The first column indicates the topic id. The second column is the Dirichlet parameter for the topic - this does not have to be included and can be parametised out. The Dirichlet parameter gives a distribution of distributions. The third columns give the words allocated to a topic.

result.txt

```
1   year-old-39 student-27 private-48 baptist-69 school-69 allegedly-48 killed-48
    pastor-69 don't-39 happened-39 george-21 sweet-95 pastor-69 atlantic-21 shore
    good-69 kids-39 atlantic-21 shores-21 christian-21 school-69 sophomore-69 arr
    related-48 felony-31 charges-48 friday-94 morning-82 shooting-82 police-82 re
    police-27 student-27 tackled-69 teacher-69 students-88 semiautomatic-31 pisto
    save-48 god-69 save-48 friends-69 family-82 boy-48 apparently-39 troubled-88
    grandfather-69 clarence-27 elliott-21 saturday-48 boy's-39 parents-48 separat
    long-82 illness-48 grandfather-69 grandson-21 fascinated-31 guns-27 boy-48 ta
    identified-82 karen-95 farley-69 wounded-82 teacher-69 year-old-27 sam-69 mar
    teacher-69 susan-88 allen-69 fled-39 room-69 marino-69 shot-48 shot-82 marino
    locked-39 door-39 opening-31 fire-39 police-27 spokesman-27 lewis-48 thurston
    thurston-82 it's-39 miracle-39 didn't-39 people-27 killed-82 police-82 chief-
    primary-39 target-31 teacher-31 classmate-69 officers-27 found-39 appeared-27
    casings-48 fourteen-69 rounds-21 fired-82 gun-39 jammed-31 thurston-82 gun-39
    thurston-82 adding-27 authorities-48 interviewed-82 adult-69 withheld-27 pend
    occurred-82 complex-27 portable-95 classrooms-88 junior-31 senior-27 high-21
    kindergarten-69 grade-31 police-27 reconstruct-39 sequence-82 events-27 resol
2
3   0   0.000   year (85) band (39) record (34) years (31) show (27)
4   1   0.000   troops (161) israel (118) army (112) iran (107) military (93)
5   2   0.000   house (172) reagan (130) president (116) bush (103) office (90)
6   3   0.000   market (75) futures (50) prices (36) trading (32) markets (31)
7   4   0.000   billion (254) budget (194) million (150) year (142) bush (122)
8   5   0.000   state (122) abortion (59) tuesday (40) reed (36) officials (32)
9   6   0.000   art (62) museum (45) home (42) mother (41) children (38)
10  7   0.000   percent (508) year (202) farmers (89) million (82) report (70)
11  8   0.000   years (54) hospital (42) good (39) day (31) people (29)
```

Figure 6: MALLET Output Data

3 Requirements

This project will involve the implementation of an application to analyse the quality and difference measures of a range of topic models produced over a document set. Described further in section 3.3, the functional and non-functional requirements in this section will cover the following tasks:

Task 1 Utilising the MALLET software package, this system will automate the production of multiple topic models over the input document set.

Task 2 Applying the topic models produced in Task 1, the system will produce a range of quality and difference measures to analyse topic variability and coherence,

Task 3 The system will use the measures developed in Task 2 to produce a visual report to explore, compare and rank the topic models produced.

All requirements in this section will follow the MoSCoW prioritisation format described below:

Acronym	**Description**
M	The system **must** have this requirement to meet the specification needs.
S	The system **should** have this requirement, however the project does not rely on its implementation.
C	The system **could** have this requirement if it does not affect the implementation of a crucial requirement.
W	The system **would** like to have this requirement, however it won't be delivered at this intial implementation.

3.1 Functional Requirements

Task 1 Creating an Automated System to Run Multiple Models at Once.

Code	**Requirement**	**Priority**
FR 1	Create a software package for automated production of multiple topic models if provided with a range or array of input	M
FR 1.1	The software interface will be a Command-Line Interface (CLI)	M
FR 1.2	There will be a range of different parameters	M
FR 1.3	There will be a different model for each parameter combination	M
FR 1.4	All models will be serializable	M
FR 1.5	The software will also compute a single model if provided with a single input	S
FR 1.6	The software configuration will be imported using an ASCII configuration text file	S
FR 1.6.1	The software configuration file will be modular and expandable	S
FR 1.7	Create a new topic modelling option as a wrapper to the existing system	C
FR 1.8	The software will compute multiple topic models in parallel	W

Task 2 Generate model difference/quality measures.

Code	Requirement	Priority
FR 2	Create software modules for a range of difference/quality measures	M
FR 2.1	A module will compute quality measures for each model	M
FR 2.2	A module will compute difference measures between each model	M
FR 2.3	The software will aggregate results from modules for all topics	M
FR 2.4	The software will generate a data file containing measures for all topics	M
FR 2.5	A module will compute log-likelihood	M
FR 2.6	A module will compute at least one perceptual difference metric	M
FR 2.6.1	A module will compute pair-wise difference	S
FR 2.6.2	A module will compute topical alignment measures	C
FR 2.7	A module will compute topic size variability	C
FR 2.8	A module will compute at least one topic coherence metric	M
FR 2.9	A module will compute cluster size variability	W

Task 3 Create a visual report of model difference/quality measures.

Code	Requirement	Priority
FR 3	Create a report contained an analytical breakdown from FR1 and FR2	M
FR 3.1	The report will import and display the quality measures computed in FR2	M
FR 3.2	The report will visualise connections between topics	M
FR 3.3	The report will include graphical representation of the topic models	S
FR 3.4	The report will include a visual comparison of the topic models	S
FR 3.5	The report will show correlations between measures computed in FR2	S
FR 3.6	The report will show the topic models used when aggregating the data in FR2	C
FR 3.7	The report will visualise topics' key words	C

3.2 Non-Functional Requirements

Task 1 Creating an Automated System to Run Multiple Models at Once.

NFR 1	**Requirement**	**Priority**
NFR 1.1	The system will utilise the MALLET java package.	M
NFR 1.2	The system will be modular to increase maintainability.	M
NFR 1.3	The system will portable.	M
NFR 1.3.1	The system will be built using a standardised java version.	M
NFR 1.4	The system will allow configuration of multiple parameters in parallel per execution.	M
NFR 1.5	The system will be scalable for future development.	M
NFR 1.5.1	The system will be executable on any hardware.	C
NFR 1.6	The system will provide extensive error reports when required.	S
NFR 1.7	The system will produce execution log reports detailing the input parameters and outputs.	C
NFR 1.8	The system will be paralysed to decrease execution time.	C

Task 2 Generate model difference/quality measures.

NFR 2	**Requirement**	**Priority**
NFR 2.1	The data file output will be in a standardised JSON format.	M
NFR 2.2	The Model Diagnostic application will be modular to allow separation of individual difference/quality measures.	M
NFR 2.3	The Model Diagnostic application will be reliable.	M
NFR 2.3.1	The application will handle all known extreme values.	M
NFR 2.3.2	The application will handle all known errors and continue execution.	M
NFR 2.3.3	The application will produce extensive error reports.	S
NFR 2.4	The Model Diagnostic application will paralyse execution of measures to increase performance.	C

Task 3 Create a visual report of model difference/quality measures.

NFR 3	**Requirement**	**Priority**
NFR 3.1	The visual report will be portable and platform independent.	M
NFR 3.1.1	The visual report will be accessible through any standard web browser.	M
NFR 3.1.2	The visual report will be accessible through any standard OS.	M
NFR 3.2	The visual report will be modular to allow easy implementation of additional graphs/logs.	M
NFR 3.3	The visual report will be developed using R, Python notebooks, Mathematica, web interfaces using libraries such as D3, or other such systems	M
NFR 3.4	The user will be able to interact with the visual report.	S
NFR 3.4.1	The interface will provide visual feedback for user interaction e.g. animations or text prompts.	C
NFR 3.4.2	The interface will remember users past interactions when the report is relaunched.	W

3.3 Task Descriptions and Success Criteria

Task 1	**Creating an Automated System to Run Multiple Models at Once**
T 1.1	At this stage, the pipeline is only capable of running one single model. In order to allow easier comparison between several models, it is desirable to include an option that will make it possible to generate multiple models automatically.
T 1.2	Ranged parameter values should be accepted by the model in order to provide thorough exploration and detailed comparison between the different models produced. Currently, only single value inputs are allowed for each of the model parameters. By being able to specify multiple values, it will allow comparison regarding the effect of changing the parameter's values across different models. The total number of models to run would be determined by the product of the number of values for every parameter. Some of the parameters include: 1. Seed Index: Type Integer - an array of 100 seeds is present in the pipeline which is used for the initialisation of the Gibbs sampler. By using the same seed it is guaranteed to produce the same models in different runs; 2. Alpha Sum: Type Double - this is the sum of all topic alpha values; 3. Symmetric Alpha: Type Boolean - this variable decides if the alpha sum should be distributed symmetrically across the topics or not; 4. Beta: Type Double - stores the value for each word beta; 5. Iterations: Type Integer - the number of times to run the Gibbs sampler; 6. Number of Topics: Type Integer - the total number of topics to be generated.
T 1.3	A common directory should be used to save all the data generated by models and all the data belonging to one model should be stored in individual sub-directories. The value of ranged parameters should be identified by the file and directory names.
T 1.4	All the models should be serialised and saved in a common directory since they will be used for further tasks.

Task 2	**Generating the Models' Difference/ Quality Measures**
T 2.1	A new module, Model Statistic or Diagnostic is to be implemented which will read and re-instantiate a set of serialised models.
T 2.2	Topic coherence, Log-Likelihood and Model Pair-wise difference are essential quality and difference measures which should be calculated for or between each model. These measures must be prioritised initially.
T 2.3	Topic Size Variability and Topic Alignment are desirable measures which should be calculated.
T 2.4	Other measures e.g. Cluster Size Variability are optional and could be computed.

Task 3	**Creating a Visual Report**
T 3.1	Data calculated from Task 2 is to be used to create a visual report. The report will help in exploring, comparing and possibly selecting topic models.
T 3.2	The report should display plots showing the models' quality and difference measures and should also present the correlations between these measures.
T 3.3	It is desirable to include interaction between the plots along with visualising the topics and their leading words.

3.4 Traceability Matrix

	Tasks										
Requirements	**1.1**	**1.2**	**1.3**	**1.4**	**2.1**	**2.2**	**2.3**	**2.4**	**3.1**	**3.2**	**3.3**
FR 1	X	X									
FR 1.1	X										
FR 1.2	X	X									
FR 1.3	X										
FR 1.4	X			X	X						
FR 1.5	X	X									
FR 1.6	X	X	X								
FR 1.6.1	X	X	X								
FR 1.7	X										
FR 1.8	X		X								
FR 2					X	X					
FR 2.1						X					
FR 2.2						X					
FR 2.3					X						
FR 2.4					X						
FR 2.5						X					
FR 2.6						X					
FR 2.6.1							X				
FR 2.6.2							X				
FR 2.7							X				
FR 2.8						X					
FR 2.9								X			
FR 3									X		
FR 3.1									X	X	
FR 3.2									X		
FR 3.3									X		X
FR 3.4									X		X
FR 3.5									X	X	
FR 3.6									X		
FR 3.7									X		X

	Tasks										
Requirements	**1.1**	**1.2**	**1.3**	**1.4**	**2.1**	**2.2**	**2.3**	**2.4**	**3.1**	**3.2**	**3.3**
NFR 1.1	X			X							
NFR 1.2	X										
NFR 1.3	X										
NFR 1.3.1	X			X							
NFR 1.4	X	X									
NFR 1.5	X	X									
NFR 1.5.1	X										
NFR 1.6	X										
NFR 1.7	X		X								
NFR 1.8	X		X								
NFR 2.1					X						
NFR 2.2						X	X	X			
NFR 2.3					X	X	X	X			
NFR 2.3.1					X						
NFR 2.3.2					X						
NFR 2.3.3					X						
NFR 2.4					X	X	X	X			
NFR 3.1									X		
NFR 3.1.1									X		
NFR 3.1.2									X		
NFR 3.2									X	X	X
NFR 3.3									X	X	X
NFR 3.4									X		X
NFR 3.4.1									X		X
NFR 3.4.2									X		X

4 Interface Specification of Serialised Model

4.1 Interface Specification

The software will be accessible using a command line interface. Upon running the software, parameters will be drawn from a local configuration file using the format specified below. If no filename is provided as an argument, a default config file named **default.conf** will be used. If only single values are given in the configuration file, a single model will be run. In the case that ranges or arrays are given a model will be run for every combination of the parameters i.e. the product of the total number of each parameter. The parameters will be printed to the display as a visual confirmation before computation begins. During the runtime, feedback on the progress will be displayed as a textual progress bar (such as the one found at `https://github.com/ctongfei/progressbar`). This will consist of the number of iterations completed over the total number of iterations over all models and the corresponding percentage.

4.2 Configuration Format

The configuration file will be an ASCII text file using the extension **.conf**. Parsing will be as per regular **.conf** files. Lines beginning with # will be ignored as comments. Lines with the format $p : n \mid n - m \mid n - m - c \mid n_1, n_2, ..., n_z$ will be interpreted and stored within a JSON object with key p. n indicates a single value, $n - m$ indicates the range from n to m - both inclusive. $n - m - c$ indicates the range from n to m with a step of c (inclusive of n, inclusive of m if the difference between n and m is a multiple of c. $n_1, n_2, ..., n_z$ is an array of specified values. An example parameter file exploring various values for alpha over the range of 100 different seeds each is given:

```
seed: 1-100
iterations: 250
alpha: 0.8, 1.6, 2.4
beta: 0.4
burning_interval: 45
optim_interval: 15
```

A **.conf** format has been chosen as opposed to using a JSON file as JSON does not support comments. Readability and ease of editing is also improved for users, since the syntax is simpler than standards like XML or JSON, without sacrificing functionality. It also more scalable than a text file and is preferred if additional parameters or other configuration settings should need to be stored in the same file.

4.3 Serialised Model

All topic models generated will be stored in a directory unique for that run, described by the datetime at which the topic models were generated. Each topic model will be serialised into a single JSON file so that they can be recreated, or deserialised, by other modules, such as *Diagnostic*. This includes, but is not limited to, the topics and documents associated with the model, output by MALLET and normally stored in the *documents* and *topics* JSON files, the parameters used to generate the model and metrics calculated during runtime such as log-likelihood. Variables (as well as attributes and methods in the source) will be stored using snake_case and classes stored using PascalCase notation. An example is given below:

```
Model{
    Topics:{
        "topic1":{
            "top_words": ["x","y","z"],
        },
        [...etc...]
    },
    Documents:{
        "1.txt":{
            "x": 0.025,
            "y": 0.032,
            "z": 0.122
        },
        [...etc...]
    },
    Parameters:{
        "iterations":200,
        "num_of_topics":20,
        [...etc...]
    },
    Metrics:{
        "log_likelihood": -7.95
    }
}
```

The parameters will also be used to generate the unique filename for the topic model. This will be done so topic model parameters can be read, compared, and evaluated without requiring each model to be opened and processed. In the case of hundreds of topic models this should result in faster execution times for related tasks, in addition to ensuring uniqueness of filenames. The parameters shall be converted into 2 or 4 digit hexadecimals, padded and concatenated for a uniform filename e.g.

```
iterations: 100 => 64_(16) => 0x0064
numOfTopics : 20 => 14_(16) => 0x14

final filename => 006414.json
```

Parameters should still be stored within the serialised model for redundancy and for the case where end users change filenames. The **default.conf** containing all parameters, that is provided with the input, will also be stored in the directory with the topic models. This will allow the encoded filename to be decoded as the parameters provided to generate the topic model is variable so cannot be decoded without a key i.e. which parameters were used to encode it.

5 Mathematical Definitions

5.1 Perceptual Difference Metrics

Perceptual difference metrics are used to relatively compare a set of topic models - contrary to quality metrics which give an objective score for each model. The insight they provide through analysis of results can include, given a set of documents:

- The similarity between two topic models
- Which topics emerge most often
- What does an average topic model look like

These insights, particularly when contrasted with results from quality metrics, will prove invaluable to the project.

5.1.1 Sum of Pair-wise Differences

Topic models can be compared by calculating the distance between them; the distance between two topic models can be found by applying a distance function to normalised vectors for each topic in both models to build a matrix of topic distances, then pairing up distances using an assignment algorithm before summing the differences between each pair. More specifically, for two topic models S_1 and S_2, with corresponding sets of topics T_1 and T_2, find the top-M-words vector $v_{i,j}$ for topic $t_j \in T_i$. The weight for word c can be given as $v_{i,j,c}$. The vectors are then normalised using formula 1

$$\hat{v} = \frac{v}{\|v\|} \tag{1}$$

The distance matrix D is calculated by applying a distance function δ over every pair i, j like $\delta(v_{1,i}, v_{2,j})$ for m topics in T_1 and n topics in T_2 to give the matrix in formula 2.

$$D = \begin{bmatrix} \delta(v_{1,1}, v_{2,1}) & \delta(v_{1,1}, v_{2,2}) & \cdots & \delta(v_{1,1}, v_{2,n}) \\ \delta(v_{1,2}, v_{2,1}) & \delta(v_{1,2}, v_{2,2}) & \cdots & \delta(v_{1,2}, v_{2,n}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(v_{1,m}, v_{2,1}) & \delta(v_{1,m}, v_{2,2}) & \cdots & \delta(v_{1,m}, v_{2,n}) \end{bmatrix} \tag{2}$$

The next step is to use an assignment strategy to pair up points in the distance matrix, using each point exactly once, creating a set of pairs P. The final pair-wise difference d is calculated as in formula 3

$$d = \sum_{\forall (p_0, p_1) \in P} |p_0 - p_1| \tag{3}$$

The application of an assignment strategy aims to minimise this value d.

Distance Functions There exist a number of formulas to calculate the distance. Examples are listed using the function δ for the distance between two vectors A, B with dimensionality n.

- Manhattan Distance (ℓ1-norm of distance)

$$\delta(A, B) = \sum_{i=1}^{n} |A_i - B_n| \tag{4}$$

- Euclidean Distance (ℓ2-norm of distance) [15]

$$\delta(A, B) = \sqrt{\sum_{i=1}^{n} (A_i - B_n)^2} \tag{5}$$

- Chebyshev Distance ($\ell\infty$-norm of distance) [16]

$$\delta(A, B) = \max_{i}^{n} |A_i - B_n| \tag{6}$$

- Cosine Distance (Angular Separation)

$$\delta(A, B) = 1 - \frac{A \cdot B}{\|A\|\|B\|} \tag{7}$$

Manhattan (4) and Euclidean (5) Distances can be generalised using Minkowski Distance with λ values of 1 and 2 respectively [17]. As λ tends to infinity, the Chebyshev Distance (6) emerges. Other metrics for distance exist such as Canberra and Bray Curtis Distances, but only the most relevant have been described.

Assignment Algorithms An assignment strategy aims to optimise the total cost of a set of pairings - in this case aiming to minimise the sum of differences in the space of all possible pairings. The problem in this case is a unbalanced assignment problem since the distance matrix isn't necessarily square. The sum of differences d will here be referred to as the cost.

An extremely inefficient example of a solution to illustrate the concept would be to calculate the cost for every possible set of pairings then perform a search to find the set with the smallest.

Another simple solution is to form pairs of the most similar values first - taking a *best-match-first* approach. This solution, however, is not global and may not lead to a set of pairings with the true minimum cost. For use within this project however, this is highly unlikely to be a concern, since matching topics is less of an exercise in reducing cost and more finding pairs of the *most similar* topics.

An alternative solution is the Hungarian (or Kuhn–Munkres) Algorithm. The algorithm is global, meaning it will always find the set of pairs with global minimum cost.

Beyond the Hungarian Algorithm, there are few established general solutions to be found, and those that do exist appear to be variations on it.

5.1.2 Topical Alignment

An alternative measure for comparing a set of models is to use **Topical Alignment** to find the *alignment* - which topics are shared across the models and how often. It is supposed that running the stochastic Gibbs sampling algorithm numerous times will eventually give the same topic(s) as a previous run, and so to measure how often this happens for a particular corpus, the alignment metric can be applied.

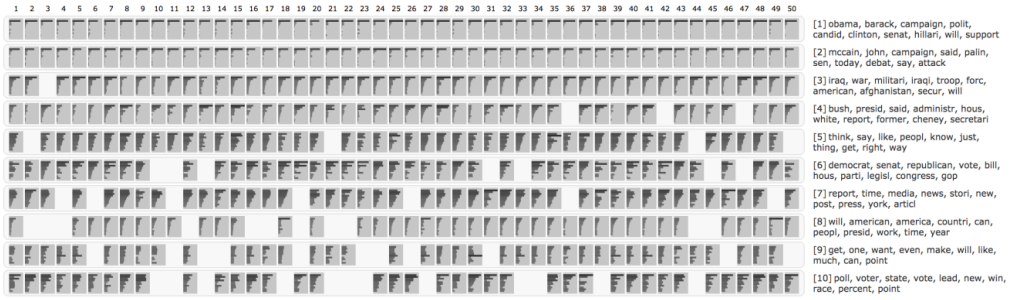

Figure 7: Documents related to specific topic groups

A visualisation is shown above[18], in Figure 7, with each rectangle being a topic, columns representing models and rows indicating which cluster the topic has been grouped with. Topics are diagrammed with bar charts showing the weighting of the top words. A blank space occurs when a topic is outwith the similarity threshold of the cluster. Observing the number of blanks for a cluster or model can give insight into how different a model could be considered to be.

5.2 Quality Metrics

5.2.1 Fitness Metrics

Fitness metrics are quantitative values used to evaluate the 'goodness of fit'. 'Goodness of fit' describes the statistical term for the discrepancies between measured results and the expected values.

Perplexity This is a measure of how accurately a model predicts a sample defined as a single quantitative value. Perplexity can be described as a measure of the "confusion" of a model. A good model should have low perplexity for a valid sample of data, such as the test set, as the model is not "perplexed" to see it. [19] For example, a model with a perplexity of 2^{20} is as confused on test data as it would be on choosing uniformly across 2^{20} possibilities. For this reason, a low perplexity is a good indication that the model is good at predicting the sample. Perplexity PP of a model q is defined on a test sample x below. The base b is often chosen to be 2.

$$PP(q) = b^{-\frac{1}{N}\sum_{i=1}^{N} \log_b q(x_i)} \tag{8}$$

Perplexity can be computed trivially without requiring access to the inner-workings of the system that generated it. [20] This would mean that perplexity can be calculated without access to the internal systems of MALLET, simply the output models.

Log-Likeliness Likeliness is a function that measures the 'goodness of fit' of a model against a sample of data. The log-likeliness is simply the natural logarithm of the likeliness function. This makes log-likelihood a common measure for quantifying the validity of a model, and can be used during the sampling process to evaluate models. MALLET automatically prints out a model's log-likelihood throughout this process, however, these printed values are lost once the sampling process is complete and so the log-likelihood can be stored in the serialised model to be retrieved afterwards for analysis. MALLET defines the log-likelihood with the following function, where "testing" is a test set:

```
double logLikelihood =
evaluator.evaluateLeftToRight(testing, 10, false, null);
```

Within the scope of this project, the assumption is that the mathematics behind log-likelihood can remain within a computational "black box" as MALLET automatically calculates the log-likelihood. This automatically output data can then be used to plot graphs and form analyses, without necessitating knowledge of the mathematical method by which MALLET calculated the log-likelihood.

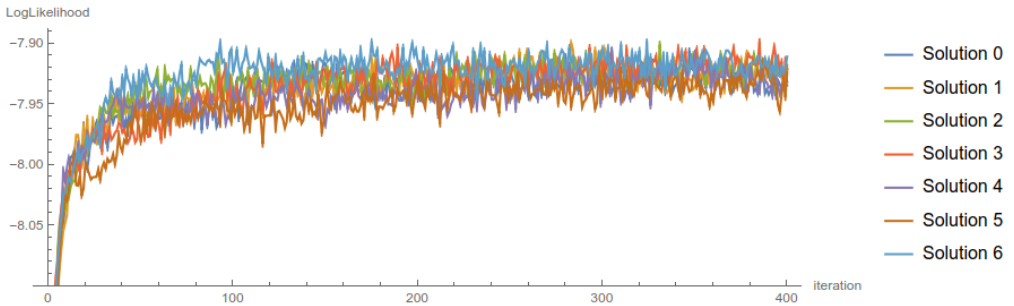

Figure 8: A composite graph of the log-likelihoods of these generated models.

These log-likelihoods can then be used to compare each model's validity against one another by comparing their average log-likelihoods, such as by taking an average over the past few hundred iterations. This provides a relatively "clean" numerical valuation of a model's validity, with high average log-likelihood values representing high model validity.

5.2.2 Coherence Metrics

The topic coherence measure is a numeric value, which attempts to evaluate the human-interpretability of a given topic.

Point-wise Mutual Information (PMI) PMI is one example of how to calculate a coherence score between two words. The higher this score, the more that a group of related words makes sense together. [21]

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \tag{9}$$

$P(w_n, w_m)$ is the number of times that two words appear together and $P(w_n)$ is the number of times that word n appears in a document.

Log Conditional Probability (LCP) This metric measures the co-occurrence of two words, which is the probability those two words appear together. LCP is very similar to PMI, however it only uses the most common word as a denominator instead of both. This minor change in the formula results in a higher correlation to human scores compared to PMI. [22]

$$LCP(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_2)} \tag{10}$$

$P(w_n, w_m)$ is the number of times that two words appear together and $P(w_n)$ is the number of times that the most common word of the pair appears in a document. [23]

The LCP coherence metric can be used by MALLET; the formula changes slightly from the LCP formula in formula 10, however the principle stays the same. In MALLET, the scores of each distinct top-ranked word are summed. This formula requires both words to each appear at least once in the document. The result is negative, and the calculated coherence of the two words increases as the result tends to zero; the closer the LCP score is to 0, the higher the coherence.

$$\sum_{i} \sum_{j<i} \log \frac{D(w_j, w_i) + \beta}{D(w_i)} \tag{11}$$

$D(w_j, w_i)$ is the total number of times the two words appeared together. $P(w_i)$ is the number of times word i (the highest ranked word of the two) appeared in the document. β must not be equal to 0 - its sole purpose is to avoid the possibility of $log(0)$ math errors.

Cosine Similarity Coefficient (CSC) This measure requires a vector space of word groups to be created. This vector space is a multi-dimensional space where each dimension is a separate word. A vector in this space is comprised of a combination of different word dimensions, depending on which words are contained within that group. The lower the CSC, the higher the similarity is between two groups of words. [24]

$$similarity = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} \tag{12}$$

A and B are vectors that represent a vocabulary. $\|A\|$ is the magnitude of a vector A.

Sorensen-Dice Coefficient (SDC) The SDC takes two groups of words using the sliding window technique, which takes around 5 words per window, and measures how much similarity there is between these groups. The greater the SDC, the higher the similarity between the the groups [24].

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{13}$$

X and Y respectively represent groups 1 and 2. $|X|$ and $|Y|$ represent the total number of words in group 1 and 2 respectively.

5.2.3 Graphical Metrics

Graphical quality metrics of topic modelling makes it easier for the users to analyse the various models produced in terms of the mapping point of view.

Topic and Cluster Size Variability A lot of the applications making use of topic modelling technique will aim to show the importance of each topic across the entire corpus of documents e.g. the topic map tool developed by the Strategic Futures Laboratory [2] shows the grouping of different topics by bubble area where the bubble size corresponds to the importance of each topic across the corpus of documents. The bigger the size of the bubble, the higher the importance it holds.

Along with that, the tool also groups similar topics together and places them in the same cluster (bubble group) to help the user with abstraction and making it easier to study the topic map.

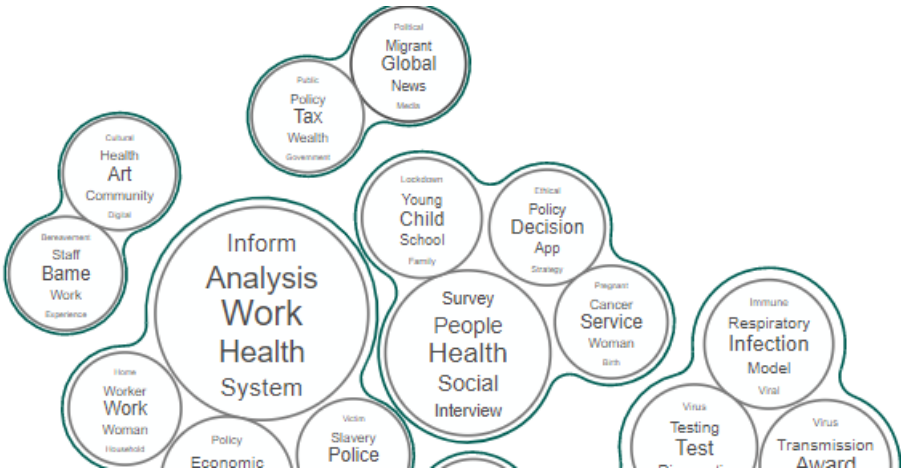

Figure 9: Visualisation of Topic Sizes [2]

In certain instances, some of the topics generated by the model will be abnormally large due to the fact that they attract a lot of the common terms from the text. This can be avoided by eliminating those terms with the help of a personalised words stop list. However, sometimes these words can be important and removing them would not be a suitable solution. One way of comparing the performance would be to examine how the different models performed with regard to their generic terms and see the difference in the sizes of the topics produced.

Agglomerative Clustering Agglomerative clustering is one of the most common hierarchical type of clustering algorithm used for grouping different objects in a cluster based on their similarity ratings. It uses the 'bottom-up' technique for this which is where the algorithm begins the process by treating each object as a single cluster and after every step pairs of clusters, which are most similar to each other, are merged together and the step is repeated until one big cluster is left which contains all the objects.

The opposite of agglomerative clustering is divisive clustering which uses the 'top-down' technique where all the objects are in one single cluster and after every step the cluster which has the least similarity between its object is divided into two and the procedure is repeated until every single object is in a cluster of its own [3].

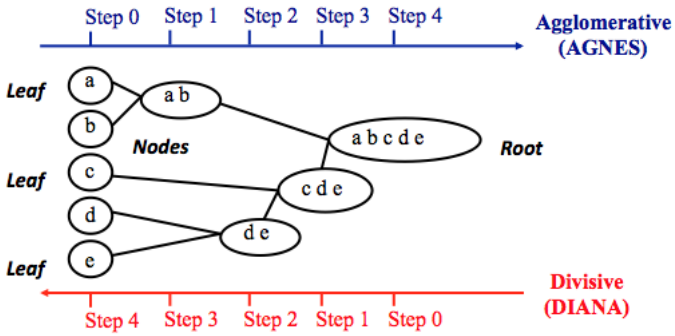

Figure 10: Hierarchical Clustering [3]

The steps used by the agglomerative clustering algorithm are as follows:

1. Each object is placed in its own cluster.
2. Distance measurement is determined and calculations are done to find the distance matrix. Euclidean Distance is one of the most used distance measurements which is calculated by formula 14.

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{14}$$

3. Linkage criteria, which defines the distance between clusters, is determined to group similar clusters together, calculated by formula 15.

$$D(X, Y) = min_{x \in X, \quad y \in Y} d_{x,y} \tag{15}$$

4. Distance matrix is updated.
5. Procedure is repeated until all the objects are in one big cluster.

```
SIMPLEHAC(d_1, ..., d_N)
 1  for n ← 1 to N
 2  do for i ← 1 to N
 3       do C[n][i] ← SIM(d_n, d_i)
 4       I[n] ← 1  (keeps track of active clusters)
 5  A ← []  (assembles clustering as a sequence of merges)
 6  for k ← 1 to N − 1
 7  do ⟨i, m⟩ ← arg max_{⟨i,m⟩:i≠m∧I[i]=1∧I[m]=1} C[i][m]
 8       A.APPEND(⟨i, m⟩)  (store merge)
 9       for j ← 1 to N
10       do C[i][j] ← SIM(i, m, j)
11            C[j][i] ← SIM(i, m, j)
12       I[m] ← 0  (deactivate cluster)
13  return A
```

Figure 11: Agglomerative Clustering Algorithm [4]

A dendrogram is produced which visualises the objects in a tree based representation. The height of the link in the dendrogram determines the distance between the objects. The information found in the dendrogram can be studied to determine the optimal number of clusters for the data by picking a suitable value for the cut-off point.

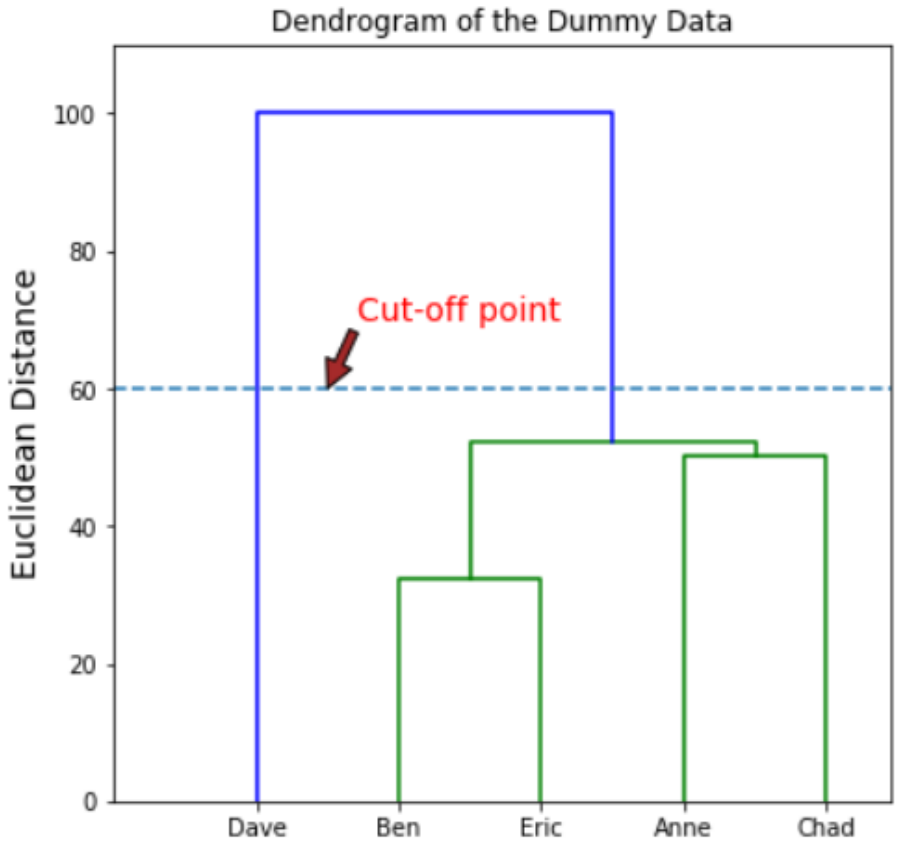

Figure 12: Agglomerative Clustering Algorithm [5]

In the dendrogram above, the cut off value has been set to 60. By setting the value to 60, two clusters are formed: (Dave) and (Ben, Eric, Anne, Chad). The number of clusters would change by setting the cut-off point to 52. We would get three different clusters: Dave, (Ben, Eric) and (Anne, Chad). By looking at the different clusters formed, it is best that the user manually decides the cut-off value which seems more optimal for the data [5].

In certain scenarios, the hierarchy displays what is known as "staircasing" which is where the topics are just grouped together in turns to the same cluster. This results in one massive cluster and many single clusters which reduces the effectiveness of the abstraction. Analysing the cluster-size variability between the different models can assist the user in selecting a model that displays the best visualisation in terms of the abstraction of the topics.

Verifying Dissimilarity In the hierarchical cluster tree, each object in the data set is connected together at some level. The link's height shows the distance between the clusters where the two object are located. The height between the objects is called the cophenetic distance. The quality of the cluster tree produced by the linkage function can be measured by comparing the cophenetic distances against the pairwise distance function which generates the original distance data.

A cophenetic correlation coefficient is generated when comparing the two values and the nearer the value is to 1, the more accurate the clustering has been done of the data. A valid clustering will have the linked objects in the cluster tree to be strongly correlated with the distances between the objects in the distance vector [6].

Verifying Consistency Comparing the height of each link against their neighbouring links below in a tree is a way of determining the natural divisions of cluster in a dataset. A relation that is of a similar height compared to the links below implies that there are no distinctions present between the objects linked at this stage of the hierarchy. These connections are said to show a high degree of continuity, since the distance between the objects being linked is roughly the same as the distance between the objects they enclose.

A link where the height is noticeably different compared to the height of links in the tree below shows that the linked objects at this stage of the cluster tree are further apart from one another when compared to their components at the point where they were initially merged. A link of such kind is known to be inconsistent to the links found below it.

The dendrogram shown in Figure 13, below, shows inconsistent links since it is joined into two groups which are merged together by links at a very high stage in the tree. The links are inconsistent when analysed against the links found at lower level in the hierarchy [6].

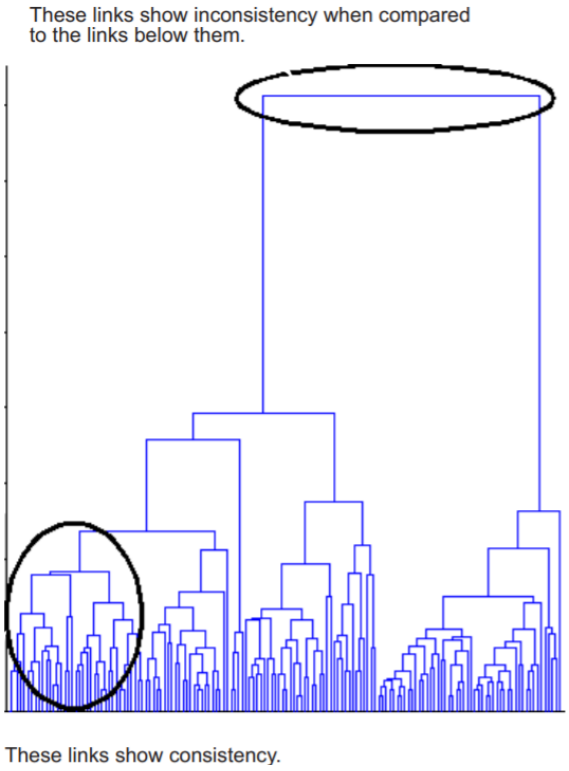

Figure 13: Showing Inconsistent Dendrogram Links [6]

k-Means Clustering k-Means Clustering works by partitioning the data into k mutually exclusive clusters and returns the cluster's index to which each object has been assigned. k-Means clustering produces a single level of clusters and uses actual observations for this process rather than using the set dissimilarity measures. This algorithm is usually more suitable for large amounts of data compared to Hierarchical Clustering [6].

Steps involved in k-Means Clustering algorithm are as follows:

1. Specify the total number of clusters to be generated.
2. Specify the total number of clusters to be generated (k).
3. Randomly pick k objects from the set of data as the initial centre for clusters.
4. Each observation is to be assigned to their closest centroid with the help of Euclidean distance between the centroid and the object.
5. For each k cluster, the centroid of the cluster is updated by generating new mean values of all data points present in the cluster.
6. Keep repeating the previous two steps until the assignments stop changing or until the maximum number of iterations has been reached. [25].

From the data generated by the output of k-means, a silhouette graph can be generated which shows the closeness of each point in the cluster to points in its neighbouring clusters. The values range from +1 to -1 where +1 indicates that points are very distant from the neighbouring clusters and -1 indicating that there is a high chance that the point has been assigned to the incorrect cluster.

The graph below shows that many points with low silhouette values are present in the first cluster and the second cluster has some points that contain negative values which tells us that the two clusters are not very well separated. The third cluster has a large silhouette value which shows that the cluster is fairly disconnected from its neighbouring clusters.

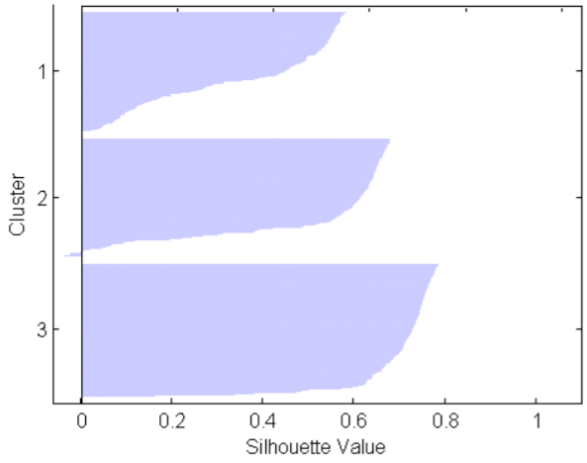

Figure 14: Showing Silhouette Values [6]

Number of clusters to be generated can be increased to see if the algorithm is able to find a better grouping and the average silhouette values can be compared between the different graphs produced to see which number of clusters are more optimal for the dataset [6].

6 Project Management

6.1 Gantt Chart

6.2 Table of Responsibilities

6.2.1 D0 Responsibilities

For the D0 deliverable, three main responsibilities were identified. These are the file structure for latex, git scripts that highlight commits and feedback on the peer review. The peer review feedback was completed by all students, the content of which was collated into one document by a designated team member. The students that took the lead for the D0 deliverable were:

D0 Responsibilities	
D0 Latex	Ridwan Mukhtar
D0 Git Scripts	Cailean Welsh
Peer Review	Mark Schmieg

6.2.2 D1 Responsibilities

The D1 deliverable was split into several subsections, each having their own lead. The following table indicates the lead for each D1 task:

D1 Responsibilities	
Background/Motivation	Ridwan Mukhtar
Functional Requirements	Matthew Frankland
Non-Functional/Constraint Requirements	Ross Moir
Interface specification of serialised model	Cailean Welsh
Mathematical Definitions of Difference and Quality Measures	George Hughes
Table of Task Descriptions and Success Criteria	Farhan Hafiz
Gantt Chart	Sabina Jędrzejczyk
Table of Responsibilities	Ridwan Mukhtar
Requirements Traceability Matrix	Mark Schmieg
Technical Risk Table	Nicolas Sparagano

Mathematical Definitions of Difference and Quality Measures was worked on by: George Hughes (lead), Nicolas Sparagano, Mark Schmieg and Farhan Hafiz.

The MALLET section in background was based on the work Cailean Welsh and Matthew Frankland did for the D1 preparation.

The final D1 LaTeX document was created and formatted by Nicolas Sparagano and Ridwan Mukhtar through the collated works of the team, as mentioned above.

6.2.3 Other Responsibilities

This table highlights the tasks that students are responsible for outwith the main project deliverables. One responsibility is to record the actions that take place during meetings and another is having a main point of contact between students and lecturers so meetings and issues can be sorted out. The students responsible for these tasks are:

Other Responsibilities	
Record "Meeting Minutes"	George Hughes
Student Point of Contact	Sabina Jędrzejczyk

6.3 Technical Risk Table

Risk Statement This is the textual name assigned to a given risk and is used as the long-hand to refer to a specific risk, with the ID being the short-hand.

Description This is a textual description that explains the risk and it's effect on the project.

Likelihood This metric rates the relative probability of a given risk occurring. This is rated on a 5-item scale from 'Very Low' to 'Very High', with 'Very Low' risks being unlikely to happen, whilst 'Very High' risks are highly probable.

Severity This metric rates the relative impact a risk would have on the project, if the risk occurs. This is rated on a 3-item scale of 'Low' to 'High'.

Mitigation This is a textual description of the methods used to mitigate the risk, lowering the chance that it affects the project in an unmanageable manner.

6.3.1 Risk Descriptions

ID	Risk Statement	Description
R1	Data Loss	Saved data and files may be lost, through either damage to storage systems or inadvertent negligence, halting the progress of a task or compromising already completed work.
R2	Unusable Model Evaluation Metrics	Certain methods of evaluating models may turn out to be inappropriate or irrelevant to the project. Beyond this, certain evaluation methods might be infeasible for their required use with the project.
R3	Algorithms Overrun	Algorithms and processes used as part of the project may exceed expected time constraints to execute, delaying or halting tasks.
R4	Internet Connectivity	Due to the entirely online nature of the project, compromised internet connectivity can completely remove team members from being able to continue contributing to the project for a short while. The risk of this is increased due to some members regularly experiencing some form of internet outage.
R5	Illness	Due to the current global pandemic, and increasing rates of infection within the UK, there is a higher-than-normal chance of team members becoming severely ill and potentially becoming incapable of progressing with their tasks.
R6	Time Constraints	Restrictions on the amount of time required to complete a task may compromise either the success of the task or it's quality.

6.3.2 Risk Mitigations

ID	Likelihood	Severity	Mitigation
R1	Low	High	All project-related data and files will be kept in online repositories where version history and backups are kept. Additionally, team members will upload new files or updates to existing files on a regular basis such that no more than a day of work is lost if a local copy of files is lost.
R2	Low	Medium	Multiple applicable methods of evaluating models will be researched to ensure a sufficient amount will suit the needs of the project.
R3	Medium	Low	Run times based on historical uses of MALLET and any evaluation algorithms will be considered to allocate sufficient time. In the event of algorithms exceeding time constraints, multiple computers can be used to run tasks in parallel in the background whilst team members continue with the project.
R4	Low	High	A team member with internet issues will use mobile data and alternative forms of connectivity to inform the rest of the team if they are prevented from continuing their assigned tasks due to the explicitly online nature of the project.
R5	Very Low	Low	Regular online meetings ensure any disruption due to illness can be mitigated through communication with the rest of the team. Additionally, team members will follow government guidelines for reducing their exposure to reduce their chance of illness.
R6	High	Medium	Project deliverables will be broken down into smaller manageable tasks, improving the chances of correctly predicting their respective time to complete. Additionally, regular team meetings where team members outline their respective task progress will ensure people who need more time can be appropriately helped.

7 Appendix

References

[1] Ankur Tomar. Topic modeling using latent dirichlet allocation(lda) and gibbs sampling explained. 2018.

[2] Strategic Futures Laboratory. Topic maps, 2020. Accessed: 21/09/2020.

[3] Datanovia. Agglomerative hierarchical clustering, Oct 2018. Accessed: 21/09/2020.

[4] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge university press, 2008.

[5] Cornellius Yudha Wijaya. Breaking down the agglomerative clustering process, Dec 2019.

[6] Cluster analysis, Aug 2013. Accessed: 23/09/2020.

[7] David M Blei and John D Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10(71):34, 2009.

[8] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211, 2019.

[9] IEEE Staff Corporate Author. Generating taxonomic terms for software bug classification by utilizing topic models based on latent dirichlet allocation. In *2013 11th International Conference on ICT and Knowledge Engineering (ICT and KE)*, 2013 11th International Conference on ICT and Knowledge Engineering (ICT & Knowledge Engineering), pages 1–5, [Place of publication not identified], 2013. IEEE.

[10] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, 2006.

[11] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577, 2008.

[12] Shawn Graham, Scott Weingart, and Ian Milligan. Getting started with topic modeling and mallet. Technical report, The Editorial Board of the Programming Historian, 2012.

[13] Mallet diagnostics, 2020.

[14] Mallet parameters, 2020.

[15] Howard Anton. *Elementary Linear Algebra*. John Wiley Sons, 1994.

[16] C. D. Cantrell. *Modern mathematical methods for physicists and engineers*. Cambridge University Press, 2000.

[17] Kardi Teknomo. Minkowski distance, 2015.

[18] Jason Chuang, Margaret E. Roberts, Brandon M. Stewart, Rebecca Weiss, Dustin Tingley, Justin Grimmer, and Jeffrey Heer. TopicCheck: Interactive alignment for assessing topic model stability. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 175–184, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[19] Chiara Campagnola. Perplexity in language models, 2020.

[20] Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. Evaluation metrics for language models. 1998.

[21] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence, 2010.

[22] David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models, 2010.

[23] Jordan Boyd-Graber, David Mimno, and David Newman. *Care and Feeding of Topic Models: Problems, Diagnostics, and Improvements*. CRC Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, Florida, 2014.

[24] Jey Han Lau, David Newman, and Timothy Baldwin. Machine reading tea leaves: Automatically evaluating topic coherenceand topic model quality, Apr 2014.

[25] K-means clustering in r: Algorithm and practical examples, Oct 2018. Accessed: 24/09/2020.