F21DG Project Report
Task Descriptions, Traceability Matrix, and the Risk Analysis

Matthew Frankland Farhan Hafiz George Hughes

Sabina Jedrzejczyk Ross Moir Ridwan Mukhtar

Mark Schmieg Nicolas Sparagano Cailean Welsh

Table of Contents

Task 1 - Utilising the Mallet software package, create a wrapper to process the production of multiple topic models over a document corpus.

Task 1 Team: Cailean Welsh and Matthew Frankland

Task 1 Description

Task	**Utilising the Mallet software package, create a wrapper to process the production of multiple topic models over a document corpus.**
T1.1	Design a TopicModel class to generate a single topic model.
T1.2	Add getter and setters in TopicModel to run model and set model parameters.
T1.3	Add methods for retrieval of data on a topic model, including topic probabilities; words and topics in a model; and distribution.
T1.4	Design a TopicGenerator class to process a .conf file, including ranged inputs, and produce derived models.
T1.5	Add a default .conf file to TopicGenerator which will be used if no .conf file is specified or specified file is invalid.
T1.6	Add default parameter values to TopicGenerator where a .conf file is specified but does not include all parameter values.
T1.7	Add validation of required parameters to TopicGenerator class where non-default used and print parameters once validated, printing default if used with a warning.
T1.8	Add a public method to TopicGenerator class which will accept a corpus and generate models based on the pre-processed configuration file.
T1.9	Serialise each topic model to JSON and store in a directory unique for that generation.
T1.10	Add a deserialisation method to turn the JSON output back into TopicModel objects.
T1.11	Add parallel computation for multiple models to reduce processing time.

Task 1 Traceability Matrix

- CW = Cailean Welsh
- MF = Matthew Frankland

Task	**Functional Requirements**								
	FR 1.1	FR 1.2	FR 1.3	FR 1.4	FR 1.5	FR 1.6	FR 1.6.1	FR 1.7	FR 1.8
T 1.1	MF		MF					MF	
T 1.2			MF						
T 1.3			MF						
T 1.4	MF					MF	MF	MF	
T 1.5		CW			CW				
T 1.6		CW			CW				
T 1.7		CW			CW				
T 1.8						CW	CW	CW	CW
T 1.9				CW					
T 1.10				CW					
T 1.11									CW

Task 1 Risk Analysis

Risk	**Task**	**Description**	**Mitigation**	**Severity**	**Likelihood**
R1	1.3	Desired output from MALLET may change as program develops	Frequent communication between T1 and T2 teams, in addition to a modular code design to allow easy changing of MALLET params.	Low	High
R2	1.9	Serialisation output may change, impacting all other parts of the project.	Make changes as infrequently as possible. Group communication if this changes and why. Encourage modular design so adapting to new serialisation design is quick and easy.	High	High
R3	1.11	Processing time may still be too great even with parallelisation. Dependent on generation and serialisation methods.	Efficient generation and parallelisation. Serialisation must also be quick. Reduce as many factors until MALLET runtime is the largest factor.	High	Medium

Task 2 - Generating the Models' Difference and Quality Measures

Task 2 Team: Nicolas Sparagano, Ridwan Mukhtar, George Hughes, Mark Schmieg

Task 2 Description

Task	Generating the Models' Difference and Quality Measures
T2.1	**Design system architecture** Generate UML class diagrams outlining the different modules of the system that are involved with Task 2.
T2.2	**Design Metric abstract class** Create a template for methods and attributes that all metric classes will share. This template will be implemented as a Java abstract class.
T2.3	**Design dummy Metric class** Create an example dummy metric class for development purposes.
T2.4	**Design Topic Model object** Plan the methods and attributes of a Topic Model object that will represent the JSON Topic Models.
T2.5	**Implement read/write of serialised JSON models to/from Topic Model object** Create methods for reading the serialised JSON into a usable form by the main program. In addition, create a method for writing data into a JSON format.
T2.6	**Implement reading Metric classes from directory** Develop a class that reads the various Metric classes found within a given directory, to enable the main program to use the methods contained in each Metric class.
T2.7	**Implement loading Metric classes and their methods from directory** Create a class that instantiates the Metric classes needed in the program. This class calls the necessary methods, to get a fully valid Topic Model JSON object, which can be passed to the T3 team for visualisation.
T2.8	**Implement iteration of Metric class methods** Create an appropriate object structure to store class objects and their related list of loaded methods. Additionally, create the iteration required to load through each Metric class' methods and invoke their necessary functionality within the main program.
T2.9	**Implement the Topic Model object** Create a java class to represent the topic models that are read from their JSON formats.
T2.10	**Implement "perplexity" metric class** Create a metric class which calculates the perplexity of a given model.
T2.11	**Implement "pairwise difference" metric class** Create a metric class, which calculates pair wise difference and returns a list of modified Topic Model JSON objects.
T2.12	**Implement "topical alignment measures" metric class** Create a metric class that calculates topical alignment and returns a list of modified Topic Model JSON objects.
T2.13	**Implement "topic size variability" metric class**

	Create a metric class, which calculates the topic size variability and adds it to a Topic Model JSON object which is then returned.
T2.14	**Implement "cluster size variability" metric class** Create a metric class that calculates cluster size variability and returns a modified Topic Model JSON object with the value in it.
T2.15	**Implement appropriate error reports for each exception within the program** Implement the program in a way, so that all exception and errors are outputted into a log file.
T2.16	**Implement data parallelism across metric calculation** Change or add a mode to the program to allow multiple metrics to be executed in parallel.
T2.17	**Test addition of a dummy metric** Implement a dummy metric to test whether the program is modular and can integrate new metrics without needing to alter the main program's code. This also includes creating erroneous metrics to test how the system responds.
T2.18	**Test extreme/edge/normal/exceptional values** Write tests to evaluate inputted values, that may raise an error, crash the program or hinder it from doing its work normally.
T2.19	**Test resource management** Tests are to be implemented to make sure the program runs fully without running out of memory, and without crashing.

Task 2 Traceability Matrix

The table below traces each subtask to the relevant requirement that is fulfilled by that subtask. The subtasks are marked with the person responsible for their completion by marking their initials in the subtask's row and using those initials to map that subtask to its relevant requirements. The key for the responsibilities is:

- NS = Nicolas Sparagano
- RM = Ridwan Mukhtar
- GH = George Hughes
- MS = Mark Schmieg

Functional Requirements

Task	**Functional Requirements**											
	FR 2	FR 2.1	FR 2.2	FR 2.3	FR 2.4	FR 2.5	FR 2.6	FR 2.6.1	FR 2.6.2	FR 2.7	FR 2.8	FR 2.9
T 2.1	GH											
T 2.2	MS											
T 2.3	MS											
T 2.4	RM											
T 2.5				RM	RM							
T 2.6	NS											
T 2.7				NS								
T 2.8		NS	NS									
T 2.9	RM											
T 2.10	NS	NS										
T 2.11	GH		GH				GH	GH				
T 2.12	MS		MS				MS		MS			
T 2.13	GH		GH							GH		
T 2.14	RM		RM								RM	RM
T 2.15												
T 2.16												
T 2.17												
T 2.18												
T 2.19												

Non-Functional Requirements

Task	Non-Functional Requirements						
	NFR 2.1	NFR 2.2	NFR 2.3	NFR 2.3.1	NFR 2.3.2	NFR 2.3.3	NFR 2.4
T 2.1							
T 2.2		MS					
T 2.3		MS					
T 2.4							
T 2.5	RM						
T 2.6		NS					
T 2.7							
T 2.8							
T 2.9	RM						
T 2.10		NS					
T 2.11		GH					
T 2.12		MS					
T 2.13		GH					
T 2.14		RM					
T 2.15				NS	NS	NS	
T 2.16							GH
T 2.17				RM	RM		
T 2.18			RM	RM	RM		
T 2.19			MS		MS		

Task 2 Risk Analysis

Risk	Task	Description	Mitigation	Severity	Likelihood
R 2.1	T2.1	Some elements of system design are not feasible with Java.	Perform feasibility testing by producing prototypes of modules with dummy implementations.	Medium	Medium
R2.2	T2.5	JSON Serialised model format is changed during the project.	All handling of JSON strings is performed in a single class, so only one part needs to be adjusted.	Medium	High
R2.3	T2.5 T2.12	Reading many models causes Java to run out of memory.	Resource management testing will be performed to find such cases.	Low	Medium
R2.4	T2.10 T2.11 T2.12 T2.13 T2.14	Implementation/testing is slowed by large numbers of serialised models with many topics and words.	Small sets of testing models will be acquired for small scale tests.	Medium	High
R2.5	T2.10 T2.11 T2.12 T2.13 T2.14	Certain metrics require specific information from models.	The metric survey will be referenced during correspondence with the sub-team in charge of serialisation to ensure necessary data is available.	High	Medium
R2.6	T2.10 T2.11 T2.12 T2.13 T2.14	Different metrics require different model counts.	The system architecture will be designed with an awareness of this.	Medium	High
R2.7	T2.16	Data parallelism is difficult or impossible to utilise within Java.	The priority for optimisation is chosen to be low.	Low	High

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

Task 3 Team: Farhan Hafiz, Ross Moir, and Sabina Jedrzejczyk

Task 3 Description

Task	Create a visual report of model difference/quality measures.
T3.1	**Use D3 to implement visual report** Use D3.js to implement a visual report that compares difference and quality measures for a set of topic models produced in Tasks 1 and 2.
T3.2	**Design visual style of report** Generate a design plan outlining the layout and visual design choices that will be followed. The visual design will be conscious of layouts/designs that would prove challenging for people with disabilities, e.g. colour blindness.
T3.2.1	**Implement visual style into the report** Implement the visual design plan utilizing CSS to produce a consistent aesthetic.
T3.2.2	**Implement visual transitions** Design and implement visual transitions that highlight data entering, updating or leaving the graphs when a user interacts with the visual report.
T3.3	**Load the serialized model JSON files** Create a JavaScript module to load/read the serialized topic model JSON files into a data array that can be consumed by the D3 visual report.
T3.4	**Display topic models on a scatter plot** Implement a scatter plot graph to display and compare the set of topic models based on the metrics calculated from Task 2.
T3.4.1	**Implement dynamic scaling and selection of metrics used in scatter plot** Create a dynamic module allowing user selection of metrics that are used to produce the scatter graphs and implement dynamic axis scaling.
T3.5	**Modularise the implementation** Implement the program in a modular format to allow additions/removal of metrics within the graphs without editing the graph code itself.
T3.6	**Implement selection of specific topic model from the scatter graph** Create a selection module that allows the user to select and highlight any topic model from the scatter graph for further inspection.
T3.6.1	**Implement selection of multiple models at once** Extend the selection module (T3.6) to allow the selection of multiple models at once to allow for a deeper comparison between models.
T3.7	**Display the metrics for a selected topic model** Create a module to display the metrics for a selected topic model.
T3.8	**Display the chosen topics and their keywords for a selected topic model** Create a module to display the topics chosen by the selected topic model.
T3.9	**Display comparison between multiple selected models using metrics/topics** Extend the modules developed in T3.7 and T3.8 to produce a comparison when multiple topic models have been selected at once.
T3.10	**Implement interactions between different graphs**

	Create a module to highlight the topic model on all other graphs when the topic model is selected on any visualisation within the report.
T3.11	**Implement cookies to remember past user interactions** Create a memory module within the report to store the users' currently selected statistics so the visualisation can restore to the same point when the report is closed and reopened.

Task 3 Traceability Matrix

- FH = Farhan Hafiz
- RM = Ross Moir
- SJ = Sabina Jedrzejczyk

Functional Requirements

Task	**Functional Requirements**							
	FR 3	FR 3.1	FR 3.2	FR 3.3	FR 3.4	FR 3.5	FR 3.6	FR 3.7
T3.1	RM	RM			RM	RM	RM	
T3.2								
T3.2.1								
T3.2.2								
T3.3		FH						
T3.4				RM	RM	RM		
T3.4.1								
T3.5								
T3.6								
T3.6.1								
T3.7	FH	FH				FH		
T3.8	SJ		SJ	SJ				SJ
T3.9	FH	FH	FH		FH	FH		FH
T3.10						RM	RM	
T3.11								

Non-Functional Requirements

Task	**Non-Functional Requirements**						
	NFR 3.1	NFR 3.1.1	NFR 3.1.2	NFR 3.3	NFR 3.4	NFR 3.4.1	NFR 3.4.2
T3.1	RM	RM	RM	RM	RM		
T3.2	SJ	SJ	SJ			SJ	
T3.2.1	SJ	SJ	SJ			SJ	
T3.2.2	SJ	SJ	SJ		SJ	SJ	
T3.3							
T3.4							
T3.4.1						RM	
T3.5	SJ	SJ	SJ	SJ			
T3.6					FH	FH	
T3.6.1					FH	FH	
T3.7							
T3.8							
T3.9							
T3.10					RM	RM	
T3.11							RM

Task 3 Risk Analysis

Risk	Task	Description	Mitigation	Severity	Likelihood
R1	All Tasks (T3.3 T3.4)	The way that the data is outputted and saved from Task 1 and Task 2 may not be suitable to work with in Task 3, resulting in being unable to create the Visual Report.	The way that the data will be outputted in Task 1 and Task 2 will be decided on early on in the development stage and agreed on by all of the students.	High	Low
R2	All Tasks (T3.1)	The Visual Report cannot be hosted on the web due to the nature of its implementation, resulting in being unable to view the model comparisons online.	The technology chosen for the development of the Visual Report such as D3 or Jupyter Notebooks will be fully investigated before any implementation begins, in order to ensure that it is supported by the web browser.	High	Low
R3	T3.2 T3.2.1 T3.2.2 T3.4 T3.6 T3.6.1 T.3.7 T3.8 T3.9 T3.10	The colours used for the Visual Report are not good for people with colour blindness e.g. if for example a colour key was used to distinguish models on the scatter plot, people with colour blindness may not be able to tell them apart, which results in poor user experience.	Colours will be properly researched and decided on when designing the look of the Visual Report in order to ensure that it is user-friendly for all of the users.	Medium	Low
R4	T3.4 T3.4.1 T3.5 T3.6 T3.6.1 T.3.7 T3.8 T3.9 T3.10	The resulting Visual Report does not allow the user to add or view their own user-defined quality and perceptual difference measures, or view any of the other pre-existing ones.	A task-lead will be assigned to ensure that the implementation of the Visual Report supports easy addition of user-defined quality and perceptual difference metrics, and that the Visual Report itself is easily interchangeable to display different scatter plots.	High	Low