User Manual

Frankland, Matthew   Hafiz, Farhan

Hughes, George Malcolm   Jędrzejczyk, Sabina   Moir, Ross

Mukhtar, Ridwan   Schmieg, Mark

Sparagano, Nicolas Ewan Giovanni   Welsh, Cailean

November 9, 2020

Contents

List of Figures

1 Overview

The User Manual gives details on how to use all 3 programs for the F21DG project. The document highlights how to use the systems, the different inputs they all take and the different ways a user can interact with the programs.

The document first looks into the Topic Model Generation and looks into how to use the jar file that is included. It goes over the different options available with the command line interface and outlines the different configurations that can be used with a configuration file.

The second section of this document goes into the Metrics program. It documents what files a user would require to run the program, the different configurations that can be used and how to create a new metric class if required.

The third section details the different ways a user can interact with the Visual Report. This explores how a user would use the graphs to analyse different metrics on the models.

For detailed documentation on the F21DG software, please consult the Design Manual.

2 Topic Model Generation

Topic Model Generator is a wrapper program over Mallet to process the production of multiple topic models given a document corpus. It is run as a Java jar file and comes bundled with all its dependencies included, for example "java -jar topicModelling-1.21-jar-with-dependencies.jar".

Once a corpus has been imported, a user can configure the Topic Model production by specifying a configuration file ('.conf') (1.2). User's can configure the alpha value; beta value; seed index; iterations; burn in interval; optimisation interval; and whether to use a symmetric alpha.

2.1 Command Line Options

Topic Model Generator has the following command line parameters:

-c,–**corpus** <arg> *corpus file path**

-p,–**parameters** <arg> *parameters.conf file path**

-d,–**defaults** <arg> *default params file path*

-m,–**maxthreads** <arg> *if flag provided, will use max available processors as model 'num of threads' for faster execution*

-n,–**numthreads** <arg> *number of threads to use per model*

-o,–**output** <arg> *path for outputted topic models*

* Required

A valid document corpus and parameter file must be provided on each program run. If a parameter is missing from the parameter file it will be taken from the provided default parameter file. If no default parameter file is provided, or the parameter is also missing from this file, a system default will be used (1.2.2).

To use Topic Model Generator multi-threaded, specify the number of threads to be used. If the 'maxthreads' flag is specified, Topic Model Generator will use the maximum number of available processors. This will override any 'numthreads' option specified.

2.2 Configuration Files

While a valid configuration file must be specified in the Command Line Options (1.1), 0-* parameters can be provided in the configuration file. The following parameters can be specified:

- **Alpha Sum** is a constant for topic model generation.
- **Beta** is a constant for topic model generation.
- **Symmetric Alpha** is a flag to use symmetric or asymmetric alpha in topic model generation.
- **Number of Topics** is the number of topics to create in each model.

- **Seed Index** is the seed used to create a pseudo-random number generator which creates the randomized test-train splits in Mallet.
- **Iterations** is the number of iterations taken to sample the corpus during topic generation.
- **Burn In Interval** is the number of iterations before optimisation starts.
- **Optimise Interval** is the number of iterations before optimisation. This optimisation allows the data to better fit the model by some topics being more prominent than others.

A valid configuration file must:

- Include only singleton, list or ranged parameters.
- Be type correct (the program will exit and an error message will be shown if any parameter is not of the correct type). Symmetric Alpha's type is Boolean, Alpha and Beta's type is Double, and all other valid parameters have type Int.

Where ranged parameters are included, a topic model is produced for each combination of the configuration file. Comments can be added to the configuration file using #comments.

2.2.1 Example Configuration File Format

```
seed_index:1000
#comment
alpha_sum:1.0
beta:2.0
iterations:20,21,23
```

2.2.2 System Default Parameters

If Topic Model Generator falls back to system defaults while producing the applicable Topic Models, the following parameters will be used:

```
alpha_sum:1.0
beta:0.01
symmetric_alpha:false
number_of_topics:30
seed_index:1000
iterations:20,21,23
burn_in_interval:1
optimise_interval:2
```

2.3 Serialised Output Structure

An output directory can be specified by the user in the Command Line Options (1.1). Where no directory is specified, a new directory is created called 'output' in the current working directory.

Within the output directory, a new directory is created for each execution with its name being the current timestamp. Topic Model Generator outputs to this directory the configuration file (including the parameters combination used) and a JSON file containing metrics for each Topic Model produced. The name of each JSON file is a unique hash on the Topic Model's ID.

2.3.1 JSON Metric File

Each JSON file includes the:

- Coherence of each topic.
- Log Likelihood of the Topic Model.
- Normalised Log Likelihood of the Topic Model.
- Weight given to each word, split by topic.
- Parameters used to produce the given Topic Model
- Topic Model's ID

3 Metrics

The Metrics program is a tool that accepts a directory of topic models which have been generated by the MALLET wrapper program as input, performing mathematical equations to extract quality and perceptual metrics for each.

3.1 Setting Up

To use the metrics program, three types of files are required. The first is a folder containing models that are generated from the MALLET wrapper. These models are required to be in serialised JSON format, meaning each should have an ID as well as topics, metrics and parameters dictionaries within them.

The second file that is required is a configuration file. This file details how the program will run and informs the Metrics program where the input models can be found, where the output models will be stored, and any parameters a metric should use.

The third is a folder containing the metrics as .class files. The Metrics program will retrieve each of these, assert its validity, and run it on each of the models that are passed in.

If the three files are available, all the loaded metrics are applied to the models and the results are appended to the metrics dictionary for the corresponding model.

3.2 Configuration Files

A configuration file contains details on where the metrics are stored, where input models can be found, and where output models will be stored. It also allows for parameters to be specified for running the loaded metrics.

3.2.1 Default parameters

- **models_in_directory** is the location where the serialised models generated by the MALLET wrapper program exist.
- **models_out_directory** is where the models should be stored after metrics have been calculated and appended to them.
- **metrics_directory** is the path to the folder that contains the metric class files.

models_in_directory, models_out_directory and metrics_directory, if not specified in the configuration file, have a default value associated with them within the program.

3.2.2 Metric Parameters

The configuration file can also be used to specify parameters that are used within the metrics themselves, allowing a user to customise where needed without needing to touch the code for a metric.

Perplexity

- **perplexity-enable** determines if the metric should be run.
  - 0 (disabled)
  - 1 (enabled)
- **perplexity-corpus_path** describes the file path of the corpus that will be applied to topics to determine their perplexity. Possible values are any full file path to an existing file, or any partial path to an existing file within the source folder of this program.
- **perplexity-log_base** determines the logarithmic base used in all calculations throughout this metric. Defaults to 2.0. Possible values are the positive real numbers greater than 0.
- **perplexity-weight_smoothing** determines the value used as the probability of words that are not contained within a topic. Defaults to 0.00001. Possible values are the positive real numbers greater than 0.

Pairwise-Difference

- **pairwise_difference-enable** determines if the metric should be run.
  - 0 (disabled)
  - 1 (enabled)

- **pairwise_difference-distance_type** determines the algorithm used to calculate the distance between two normalised topic vectors. Defaults to *manhattan*. Possible values are:
  - *manhattan* (defined as $\delta(A, B) = \sum_{i=1}^{n} |A_i - B_n|$)
  - *euclidean* (defined as $\delta(A, B) = \sqrt{\sum_{i=1}^{n} (A_i - B_n)^2}$)
  - *chebyshev* (defined as $\delta(A, B) = \max_i^n |A_i - B_n|$)
  - *cosine* (defined as $\delta(A, B) = 1 - \frac{A \cdot B}{\|A\| \|B\|}$)
- **pairwise_difference-top_M** determines how many of the highest weights to consider when calculating the distances between topics. Defaults to 10. Possible values are the positive whole numbers.

Topical Alignment

- **topical_alignment-enable** determines if the metric should be run.
  - 0 (disabled)
  - 1 (enabled)
- **topical_alignment-top_M** determines how many weights there should be in a topic. The words with the highest weights are picked and the highest M words in a topic are stored. The default value used is 3.

3.3 Creating New Metrics

* This section is intended for advanced users

The metric program is designed to be dynamic; a user can create their own metric class in Java and have it run on the topic models, appending results into the metrics dictionary. A metric template was created to help users understand how to create their own metric easily. This is called *MetricTemplate.java* and can be found in the *src* folder of the program.

To create a new metric, first create a Java class and ensure that it extends *AbstractMetric*. By doing this, a calculate, *getMetricName* and *loadParameters* method must be implemented.

The calculate method is designed to deal with taking in the topic models and calculating the relevant metric based on the details a model contains.

Topic Models There are 4 main components to each model that can be used for calculating a metric. These are:

- **parameters** are the values used during the topic model generation stage.
- **metrics** contains the metrics calculated in MALLET. These include coherence and log-likelihood.
- **topics** is a dictionary that contains topics. Each topic has a list of words with the count of words alongside them.
- **ID** is the ID given to the model.

To retrieve the values from the topic models, getter methods are used. These are *getMetrics*, *getTopics*, *getParameters* and *getID*. Each of these return an object where metrics, topics and parameters are maps and the ID is a string. (It should be noted that the parameters for a topic model correspond to the parameters that were given to MALLET upon generation and are not related to the parameters derived from the Cofiguration object here).

To add the calculated value to the metrics in a model, you use the command:

model.addMetrics(NAME, VALUE)

Where NAME is the name of the metric and VALUE is a string, number or map that is associated with the metric. This creates a map structure where the name is the key.

Writing to an alternative location

In the event that a metrics results is based on all models being compared, use the writeFile method to store it separately if the user doesn't want to have it stored within all the models. This feature is offered since a metric may not represent the individual model but the collection of the models as a whole. The command is a protected method and can be ran in any class that extends *AbstractMethod*. You can use the command by:

writeFile(OBJECT, CONFIG_ OBJECT)

The OBJECT can be a string, number or map and is converted into JSON before writing to a file. The file is stored alongwith the models and uses the getMetricName() method to name the file that is generated.

Configuration Files
If a user requires parameters that can be changed via the configuration file, the *loadParameters* method must be implemented to load values from the Configuration object and assigned to global variables within the new metric class. Some helper methods exist for this:

Retrieving values of a specific type is done by using one of the following:

loadParameterString(config, PARAMETER_ID, "default")
loadParameterDouble(config, PARAMETER_ID, 1.0)
loadParameterInteger(config, PARAMETER_ID, 1)
loadParameterBoolean(config, PARAMETER_ID, true)

These lines of code allows for a value to be retrieved from the configuration file. These should be executed within *loadParameters*.

- *config* is of type *Configuration* and is automatically passed into the *calculate* method.
- *PARAMETER_ID* is the name of the parameter in the configuration file; the name of the metric will automatically be appended to this ID so as to allow different metrics to use different values for parameters associated with the same ID. As an example, pairwise difference and topical alignment both require a top-M parameters, but they can be distinguished by using "pairwise_difference-top_M" and "topic_alignment-top_M".
- The final argument is the default value to be used if no value is found for the given parameter ID in the configuration file - if this is the case, the method will return this default value.

Please note that there is no way for the metric parameters used to be recovered from the serialised models after running.

Installation
When dealing with the source code in an IDE, new metrics can be added by simply adding their respective .class files to the "metrics" folder. These metrics will be automatically loaded by MetricLoader.class at runtime.

When dealing with compiled code, new metrics can be added by adding their respective .class files to the "metrics" folder that is in the same directory as the .jar file (MetricLoader.jar). The metrics' .class files also need to be added directly into the .jar file. This can be done by opening the .jar using any zip file software, whereby the .class files can be copied directly into the .jar file.

4 Visual Report

The Visual Report deals with visualising the difference and quality measures which have been calculated with the Metric program. It allows a user to plot topic models given two singleton valued metrics.

4.1 Selecting the File Directory

After running Mallet, calculating the difference and quality measures, and specifying the Visual Report directory to which the model data will be written to, the user can select the desired directory from the *Choose the model directory* drop-down menu:

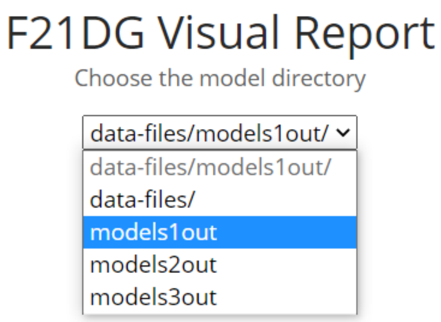


Figure 1: Choosing directory containing models

The data from the chosen directory will be then read in and displayed on the scatter plots:

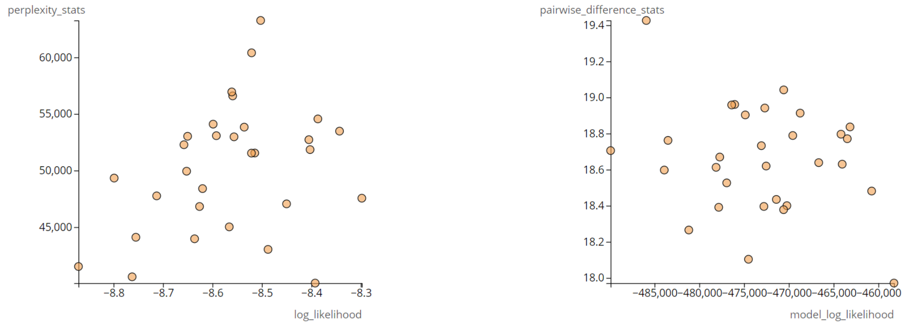
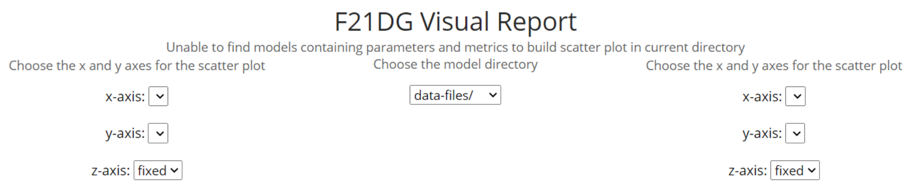


Figure 2: Displaying data on scatter plots

Note here that the "default" directory that is displayed is the *data-files* directory, which should not contain any files but instead folders with the models inside of them. On Visual Report start up the following message will be displayed:


F21DG Visual Report

Unable to find models containing parameters and metrics to build scatter plot in current directory

Choose the x and y axes for the scatter plot

x-axis:

y-axis:

z-axis: fixed

Choose the model directory

data-files/

Choose the x and y axes for the scatter plot

x-axis:

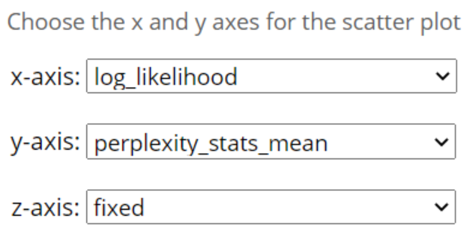
y-axis:

z-axis: fixed

Figure 3: Visual report when data-files is selected

4.2 Selecting the Scatter Plot Axes

The axes on the scatter plot can be changed to display different metrics by selecting different options from the available drop-down menus:


Choose the x and y axes for the scatter plot

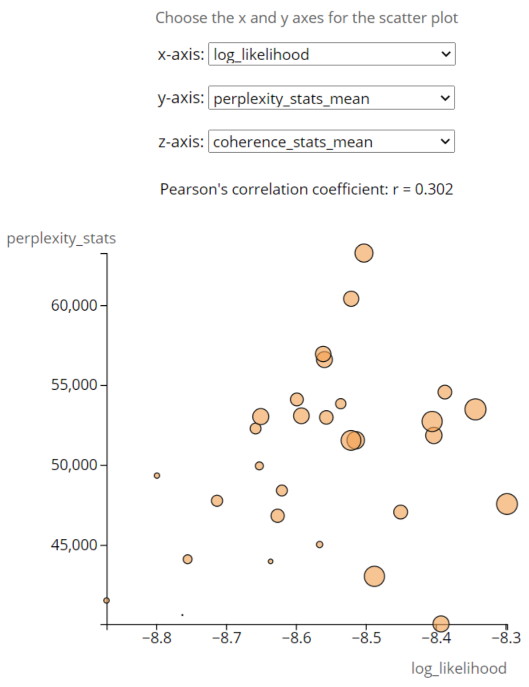
x-axis: log_likelihood

y-axis: perplexity_stats_mean

z-axis: fixed

Figure 4: Changing the x and y axis metrics

The x-axis and y-axis drop-downs are used for displaying the metrics on their respective axes, while the z-axis is used for changing the size of the circle points to be the chosen metric, on the scatter plot e.g. if *coherence_ stats_ mean* is chosen in the third axis then the circle points on the scatter plot will be resized to show the metric:


Choose the x and y axes for the scatter plot

x-axis: log_likelihood

y-axis: perplexity_stats_mean

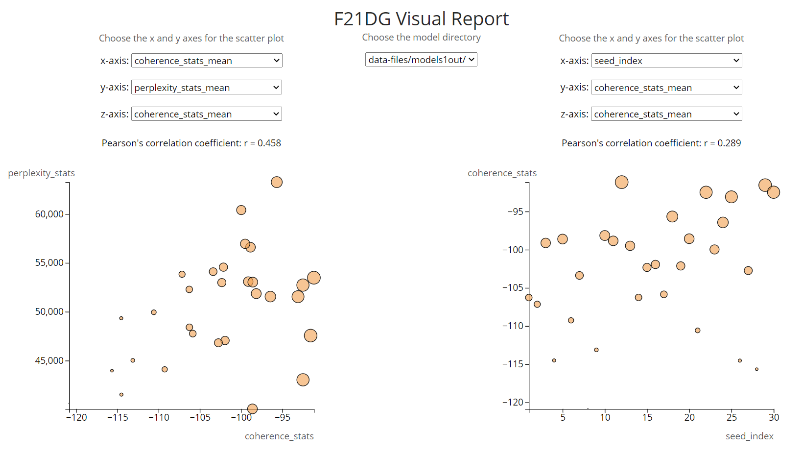
z-axis: coherence_stats_mean

Pearson's correlation coefficient: r = 0.302

Figure 5: Scatter plot showing circle points resizing after choosing coherence_stat_mean

As a default, the axes on the first scatter plot are set to show the first metric loaded in for the x-axis (in this case log_likelihood) and the third metric loaded in for the y-axis (in this case perplexity_stats_mean), While for the second scatter plot the x-axis is set to the second metric loaded in (in this case model_log_likelihood) and the y-axis is set to the fourth metric (in this case pairwise_difference_stats_mean). In both of the scatter plot cases, the third axis is set to a fixed point.

Changing any of the axes for a given scatter plot will result in an immediate change to what is displayed on the scatter plot:


F21DG Visual Report

Choose the x and y axes for the scatter plot

x-axis: coherence_stats_mean

y-axis: perplexity_stats_mean

z-axis: coherence_stats_mean

Pearson's correlation coefficient: r = 0.458

Choose the model directory

data-files/models1out/

Choose the x and y axes for the scatter plot

x-axis: seed_index

y-axis: coherence_stats_mean

z-axis: coherence_stats_mean

Pearson's correlation coefficient: r = 0.289

Figure 6: Showing the updating of the scatter plots when choosing new parameters

As the scatter plot changes, so does the calculated Pearson's correlation coefficient which shows the correlation between the 2 chosen metrics.

4.3 Selecting Models from the Scatter Plot

The data for each of the models is displayed on the scatter plot in the form of circle points:

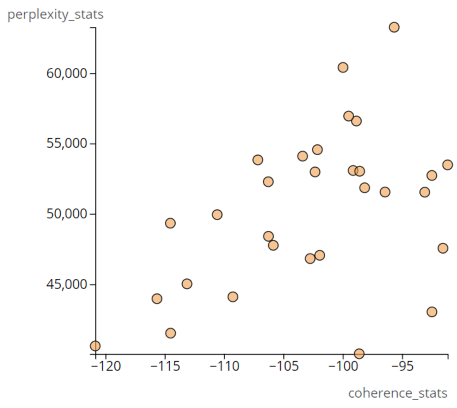


Figure 7: Displaying each model on the scatter plot

Hovering over a point on one of the scatter plots also shows that point on the other scatter plot:

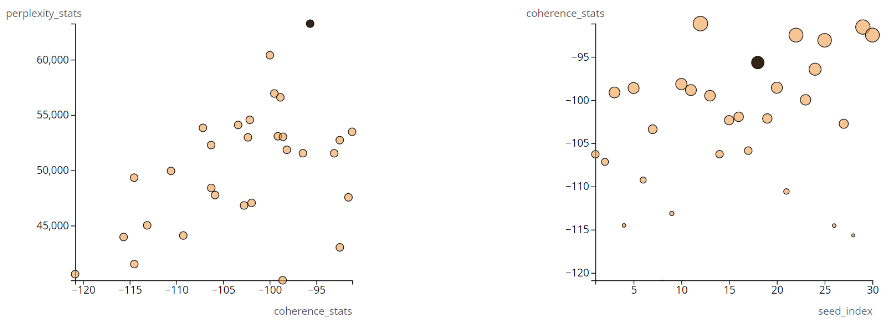


Figure 8: Showing the same point on different scatter plots after hovering over a point

To select a model, just click on a point in the scatter plot. The selection may take a while as the action of clicking on a point in the scatter plot loads in the data for that particular model. Following the selection, the following is displayed on the plot:

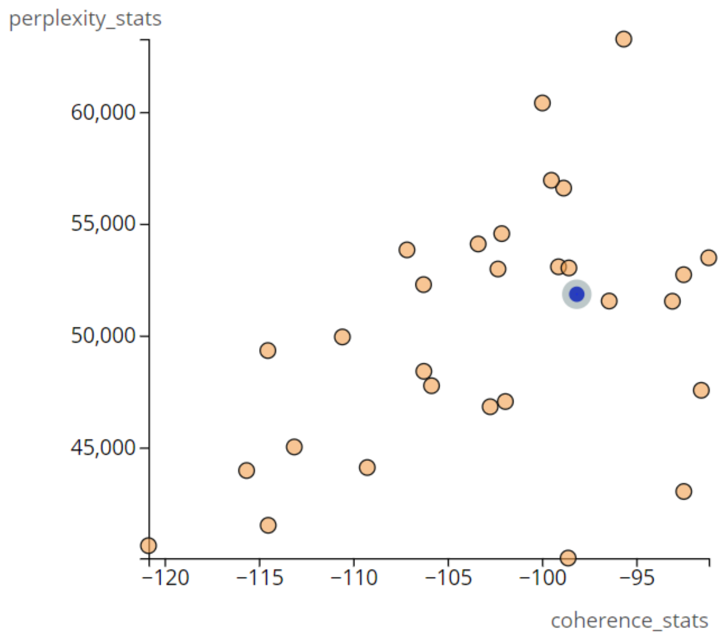


Figure 9: Showing a selected point on the scatter plot

Up to three models can be selected at one time, with the following settings: the blue selection is the selected model 1, the red selection is the selected model 2, and the green selection is the selected model 3:

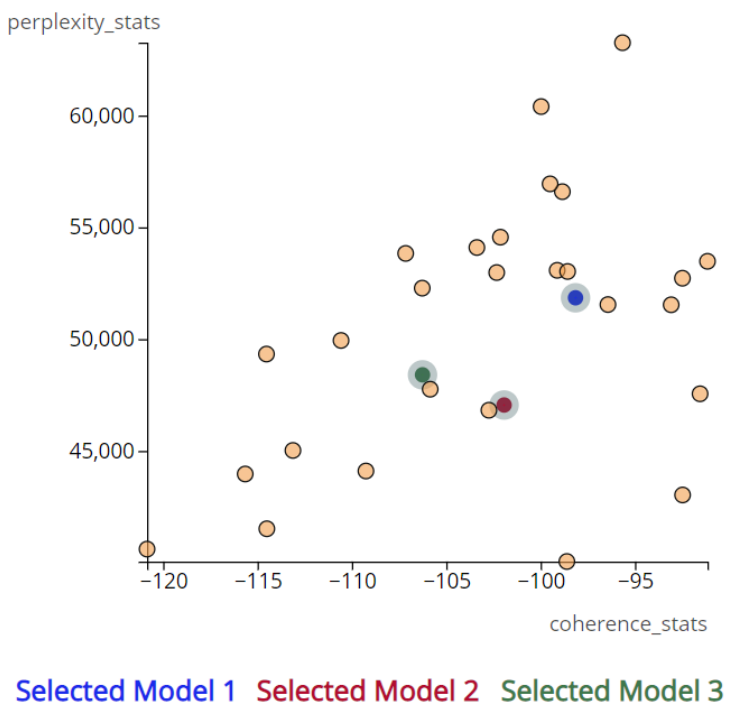
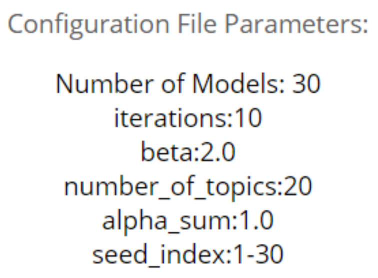


Figure 10: Showing the selection of 3 points within a scatter plot

4.4 Viewing the Model Data

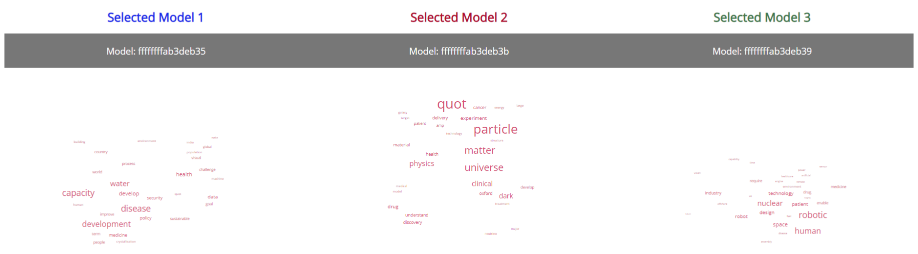
The Visual Report displays the parameter file, which was used in order to generate the models, is shown below the scatter plots:


Configuration File Parameters:

Number of Models: 30
iterations:10
beta:2.0
number_of_topics:20
alpha_sum:1.0
seed_index:1-30

Figure 11: Display of the configuration file parameters

As mentioned in the above section, clicking on a point on the scatter plot loads the data for that chosen model and displays it below the parameter file data, in the following format:

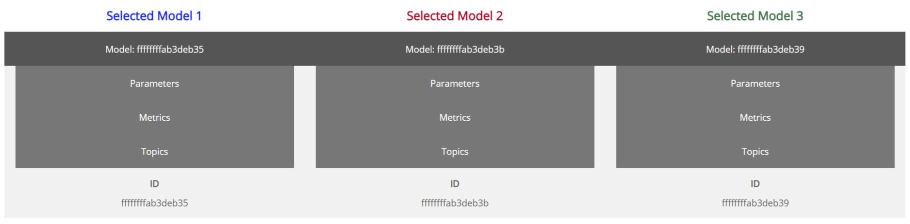

Selected Model 1 | Selected Model 2 | Selected Model 3

Model: ffffffffab3deb35 | Model: ffffffffab3deb3b | Model: ffffffffab3deb39

Figure 12: Display of the word bubbles

The word bubbles for that model are also displayed, below the main model data.

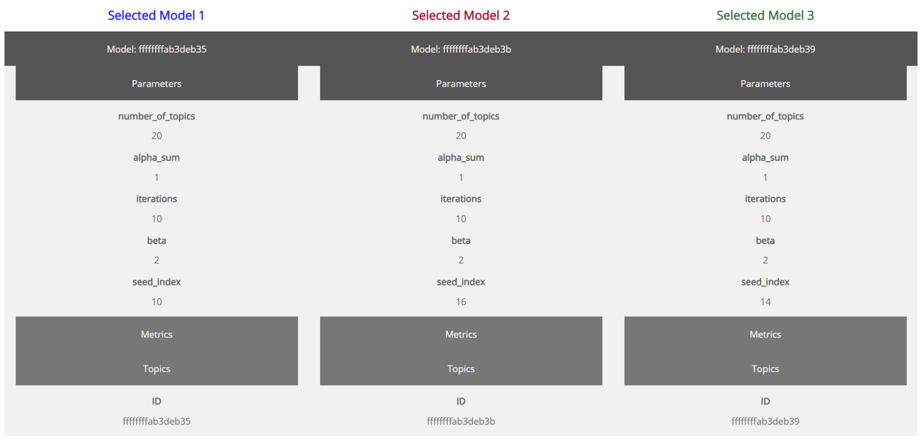
To view the data for the selected models, click on the button with the title *Model* followed by the model ID, this will expand to show more menus such as *Parameters*, *Metrics*, *Topics*:



Selected Model 1	Selected Model 2	Selected Model 3
Model: ffffffffab3deb35	Model: ffffffffab3deb3b	Model: ffffffffab3deb39
Parameters	Parameters	Parameters
Metrics	Metrics	Metrics
Topics	Topics	Topics
ID	ID	ID
ffffffffab3deb35	ffffffffab3deb3b	ffffffffab3deb39

Figure 13: Viewing the data for the selected models

To view any of the other options click on its respective name, for example clicking on *Parameters* will expand further to show a list of parameters that was used for generating that particular model:



Selected Model 1	Selected Model 2	Selected Model 3
Model: ffffffffab3deb35	Model: ffffffffab3deb3b	Model: ffffffffab3deb39
Parameters	Parameters	Parameters
number_of_topics	number_of_topics	number_of_topics
20	20	20
alpha_sum	alpha_sum	alpha_sum
1	1	1
iterations	iterations	iterations
10	10	10
beta	beta	beta
2	2	2
seed_index	seed_index	seed_index
10	16	14
Metrics	Metrics	Metrics
Topics	Topics	Topics
ID	ID	ID
ffffffffab3deb35	ffffffffab3deb3b	ffffffffab3deb39

Figure 14: Viewing further options of parameters