

# Do UK Petitions Persuade Debates on New Topics?

A dissertation submitted in partial fulfilment of  
the requirements for the degree of  
BACHELOR OF SCIENCE in Computer Science  
in  
The Queen's University of Belfast  
by  
Donal Mallon  
29/04/2019

**SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING and COMPUTER  
SCIENCE**

**CSC3002 – COMPUTER SCIENCE PROJECT**

**Dissertation Cover Sheet**

Student Name: Donal Mallon Student Number: 40154387  
Title: Do UK Petitions Persuade Debates on New Topics?  
  
Supervisor: Dr Deepak Padmanabhan

**Declaration of Academic Integrity**

Before submitting your dissertation please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.

**By submitting your dissertation you declare that you have completed the tutorial on plagiarism at <http://www.qub.ac.uk/cite2write/introduction5.html> and are aware that it is an academic offence to plagiarise. You declare that the submission is your own original work. No part of it has been submitted for any other assignment and you have acknowledged all written and electronic sources used.**

*Student's signature*

Donal Mallon

*Date of submission*

29/04/2019

## **Acknowledgements**

I would like to thank Dr Deepak Padmanabhan my Project Supervisor. Deepak provided guidance throughout my final year project and was always available to help when needed.

## **Abstract**

The aim of e-petitions in the UK is to provide a vehicle for the public to express its concerns and to call for action from the government or House of Commons. When a petition gathers enough signatures it is almost guaranteed to be debated in parliament. The aim of my study is to discover to what extent the UK petitions system has inspired debates on new topics using topic modelling. Initially, data of the wording used in petitions and transcripts of more general debates in parliament was collected through web scraping. Topic modelling was then used to compare and contrast the content of these sets of text data to discover latent patterns in the text data and thus infer topics within the text. Furthermore, a program was developed that would allow a user to explore and visualise the latent topic distributions of the two sets of documents in an attempt to give an insight into which petitions topics had been sufficiently debated in Parliament and which had not.

## **Contents**

<b>1.</b>	<b>Introduction and Problem Area</b>	<b>5</b>
<b>2.</b>	<b>Solution Description and System Requirements</b>	<b>6</b>
<b>3.</b>	<b>Data Collection System</b>	<b>7</b>
3.1.	Design	7
3.2.	Implementation	10
3.3.	Testing and validating collected data	16
<b>4.</b>	<b>Topic Modelling</b>	<b>19</b>
4.1.	LDA	19
4.2.	Implementation	19
4.3.	Analysis of models	22
<b>5.</b>	<b>Topic Explorer</b>	<b>25</b>
5.1.	Design	25
5.2.	Implementation	27
<b>6.</b>	<b>Conclusion</b>	<b>31</b>
6.1.	Project review	31
6.2.	Limitations and Possible Improvements	31
6.3.	Future Study	32
	<b>References</b>	<b>33</b>

# 1. Introduction and Problem Area

“Petition systems provide a recognised process which link citizen and state” [2, p481].

A petition is a call for change. It is created by an author and signed by other citizens sharing the same point of view. The UK Petitions system is described by the UK government as “an easy way for you to influence government policy...” [3].

However, the extent of the influence of UK petitions systems has been under scrutiny ever since the process was first devised, with many believing that the petitions system is weak and ineffective [4]. Petitioning authority can be traced back to the 13<sup>th</sup> century, and while considered as a method of delivering citizens pleas to parliament, the mechanism is believed to have had limited success [5]. However, some authors contend that petitions have the “capacity to be a main driver in expanding and deepening participative democracy” [6, p140]. In a study evaluating the effect of petitions systems, successful systems were found to have positive benefits in promoting citizen engagement in democratic politics [7].

Advances in technology and the availability of the internet has resulted in the development of E-petitions. These are aimed at making petitioning more accessible to a broader range of people. E-petitioning has become a new and popular channel for the public to have their thoughts and concerns presented to the government. The original UK online petitions process was formed in November 2006 and since the process has received several adjustments and revisions. The Parliaments petitions website was re-launched in its current form in July 2015. As of 2011, all petitions that receive over 10,000 signatures will receive a response from the government. All petitions that receive over 100,000 signatures are almost guaranteed to be debated in parliament. However, there are a small number of petitions that will receive over this threshold and not be debated on the floor such as joke petitions. To date of the 18,974 petitions submitted during the current Conservative Party minority government, only 59 petitions were directly debated in the House of Commons.

Most popular petitions, even if debated in parliament, do not reach a successful outcome [8]. The aims of this study are twofold; to provide an insight into the impact petitions have on debates in the UK Parliament and to make the petition system more transparent by devising a tool for discovering which petitions topics have been debated in Parliament and which have not.

## 2. Solution Description

The solution devised is split into three parts:

- Data Collection (Section 3)
- Topic Modelling (Section 4)
- Topic Explorer GUI (Section 5)

### **Data Collection (Section 3)**

The first part of the project is the initial collection of data from the petitions website <https://petition.parliament.uk>. The data of interest is the text data in the information about the petitions that was provided by the author. The petitions available on the UK petitions website [9] are available from July 2011 up to current petitions, therefore the project collected the information on all closed petitions since that date.

The second part of the data collection system will collect the text data available from the transcripts of general debates in parliament over the same time period. This will be achieved through web crawling the Hansard Parliament website [10] and gathering URL links to transcripts of debates in the House of Commons. From these URL links the system will then the text data of the words spoken in each debate.

### **Topic Modelling (Section 4)**

Comparing the topics that had been petitioned with the topics that had been discussed in Parliament over a similar time frame will provide an insight into the effectiveness of UK Petitions system, as well as providing an insight into the extent to which a certain topic was discussed in Parliament. Therefore, two topic models will be trained. One model on the details provided about petitions collected from the UK Petitions website [9] and the other on the UK Parliament debates collected over the same time period from the Hansard Parliament website [10].

### **Topic Explorer GUI (Section 5)**

The LDA model will produce a topic distribution for each of the models. Using the two models a GUI could be developed that would allow the user to explore and visualise the topics pertaining to petitions and debates enabling the user to compare the two distributions of topics. This program would give an insight into which petitions topics had been sufficiently debated in Parliament and which had not.

### 3. Data Collection System

#### 3.1 Design

The data collection system will be made up of two parts. The first part, to gather information on petitions that are available on the governments petitions site [9]. The second part of the data collection process, to gather the text data available from the transcripts of general debates in parliament available on the Hansard website [10].

#### Petitions Parliament Data Collection

All petitions submitted during the current Conservative minority government are available on the parliament petitions website. These petitions fall into categories such as ‘Open petitions’, ‘Closed Petitions’ and ‘Debated in Parliament’. As well as petitions submitted during the current government, the website contains archived petitions submitted during the previous two governments: 2015-2017 Conservative government and 2010-2015 Conservative – Liberal Democrat coalition government.

The UK Parliament Petitions website [9] provides data for each petition in CSV format. The petitions of interest for data collection fall into the following three categories:

- Closed petitions submitted during current Conservative minority government 2017-now.
- All published petitions submitted during the 2015-17 Conservative government
- All published petitions submitted during the 2010-15 Conservative – Liberal Democrat coalition government.

Each CSV file provides the following 4 columns of data for each petition in the category as shown in Figure 1:

- Petition: the petition title
- URL: the URL link to each petition within the UK petitions website
- State: whether the petition is open or closed
- Signature Count

	A	B	C	D
	Petition	URL	State	Signatures Count
1	Petition			
2	Introduce a rate increase cap on pay TV pricing for football	https://petition.parliament.uk/archived/petitions/104309	closed	22
3	Impose a heavy extra tax on foreign buyers of property over £3.5 million pounds.	https://petition.parliament.uk/archived/petitions/104311	closed	383
4	Hold a referendum on electoral reform with the format used in New Zealand.	https://petition.parliament.uk/archived/petitions/104317	closed	4767
5	Make the 'Steam' refund policy the law for all video game digital distribution.	https://petition.parliament.uk/archived/petitions/104318	closed	94
6	Ban unpaid internships	https://petition.parliament.uk/archived/petitions/104319	closed	438
7	Maintain current level of BBC funding as a minimum.	https://petition.parliament.uk/archived/petitions/104320	closed	77
8	Create a Proportional Representation system for UK General Elections.	https://petition.parliament.uk/archived/petitions/104324	closed	2535
9	Ban immigration by cousin marriage	https://petition.parliament.uk/archived/petitions/104325	closed	181
10	Review ECV guidelines to lower the risk of our daughters' death being repeated	https://petition.parliament.uk/archived/petitions/104329	closed	875
11	To debate a vote of no confidence in Health Secretary the Right Hon Jeremy Hunt	https://petition.parliament.uk/archived/petitions/104334	closed	231136
12	Government should officially announce beginning and end of lunar calendar months	https://petition.parliament.uk/archived/petitions/104337	closed	12
13	To stop fixed penalty notices for unauthorised absence from school	https://petition.parliament.uk/archived/petitions/104338	closed	3650
14	Introduce a UK National Day to celebrate our United Kingdom.	https://petition.parliament.uk/archived/petitions/104343	closed	10221
15	Make casinos check all visitors against a database to protect problem gamblers.	https://petition.parliament.uk/archived/petitions/104345	closed	50
16	Add more higher rate council tax bands for properties worth over £1 million.	https://petition.parliament.uk/archived/petitions/104346	closed	123
17	Make the production, sale and use of cannabis legal.	https://petition.parliament.uk/archived/petitions/104349	closed	236995
18	Scrap the House of Lords and replace with a new 'House of Heroes'	https://petition.parliament.uk/archived/petitions/104352	closed	100
19	Stop unfair tax credit changes to middle to low income earners	https://petition.parliament.uk/archived/petitions/104356	closed	2169
20	Save Vulcan XH558. Persuade technical authorities to reconsider ending support.	https://petition.parliament.uk/archived/petitions/104358	closed	11409
21	Bring back rent control (The Fair Rent Act)	https://petition.parliament.uk/archived/petitions/104359	closed	542
22	Legislate to introduce minimum sentences for assaulting police officers.	https://petition.parliament.uk/archived/petitions/104360	closed	18741
23	Keep Sunday Trading Laws the same.	https://petition.parliament.uk/archived/petitions/104363	closed	545
24	Abolish the car tax system as it is. Add it to petrol, help save the environment	https://petition.parliament.uk/archived/petitions/104364	closed	191
25	Minimum wage for carers £9 per hour we buy our own cars ,petrol,tyres etc	https://petition.parliament.uk/archived/petitions/104366	closed	9120
26	Abolish the TV licence	https://petition.parliament.uk/archived/petitions/104367	closed	14288
27	Give better pay and support for us carers	https://petition.parliament.uk/archived/petitions/104368	closed	3597

Figure 1 – Screenshot of CSV file downloaded from the petitions website [9]

The data collection system could then iterate through each URL in the CSV files provided and scrape relevant data. Each URL addresses the webpage for each petition.

The first two pieces of data scraped from each webpage are:

- Information- This is the details about the petition provided by the author of the petition
- More Information – This is the additional details about the petition provided by the author.

Petition

## Scrap the £35k threshold for non-EU citizens settling in the UK

In April (2016) the Home Office and Theresa May are introducing a pay threshold for people to remain here, after already working here for 5 years. This only affects non-EU citizens that earn under £35,000 a year, which unfairly discriminates against charity workers, nurses, students and others.

▼ More details

This ridiculous measure is only going to affect 40,000 people who have already been living and working in the UK for 5 years, contributing to our culture and economy. It will drive more workers from the NHS and people from their families. This empty gesture will barely affect the immigration statistics. It's a waste of time, money and lives.

This is the first time the UK has discriminated against low-earners. £35k is an unreasonably high threshold. The UK will lose thousands of skilled workers.

Information

More information

Figure 2 – A screenshot of one of the petitions webpages available on the UK petitions website [9]. The information and more information paragraphs are highlighted.



The third piece of data that would be collected for each petition would be the transcript of the debate if a petition had reached the threshold for debate in Parliament. If a petition had been debated in parliament the petition webpage would include hyperlink called ‘Read Transcript’. This hyperlink would address the webpage of the Hansard Parliament website that contained the verbatim transcript of the debate in the House of Commons about said debate. The Hansard petition debate webpage could then be scraped to obtain a string of all words spoken in the debate.

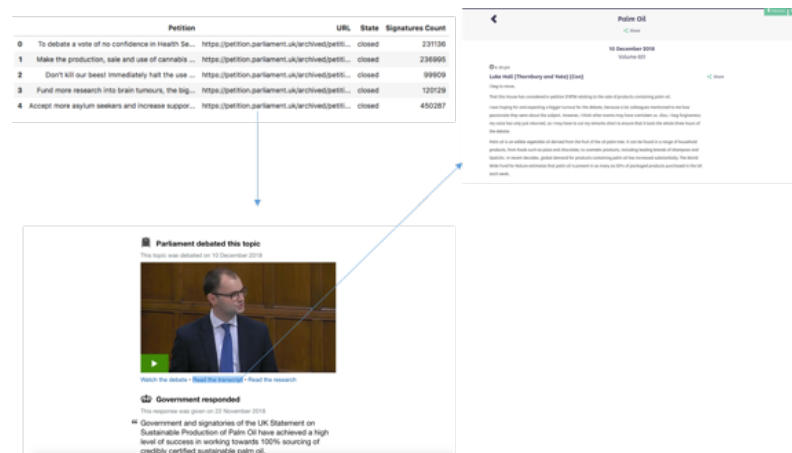


Figure 3- A diagram outlining the process for finding a transcript of a debated petition.

As a result of the information available on each petition webpage, the data collection system will therefore go to each URL and scrape the additional information as described above. Combining the new gained information with the CSV files would result in a final data frame with the following information about each petition:

- **Petition:** the petition title
- **URL:** the URL link to each petition within the UK petitions website
- **State:** whether the petition is open or closed
- **Signature Count:** number of members of the public that had signed the petition
- **information:** this was the information about the petition provided by the author of the petition
- **more information:** this was additional information provided by the author of the petition and was not always available
- **transcript:** if a petition had led to a debate in Parliament this would hold the transcript of the debate in text form.

### General Debates Data Collection

All debates in both the House of Commons and House of Lords are transcribed and made available on the Hansard website [10]. The system would collect transcripts of all debates in the House of Commons since August 2011 to December 2018 in an attempt to align them with the time frame of the petitions data.

The processes for achieving this would involve the following two steps:

- Collection of a list of all URL links to debates in the House of Commons during the stated timeframe. The Hansard website [10] provides a ‘Find Debates’ feature that allows a user to search through all debates that had occurred in the House of Commons. The feature allows the choice of a time frame for searching through debates.
- Scrape the webpage at each of these links to obtain a string containing all spoken words in each debate.

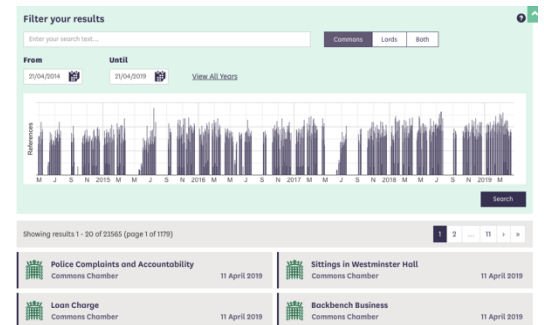


Figure 4 – Find Debates feature available on the Hansard Parliament Website [10]

## 3.2 Implementation

### Development Environment - JupyterLab

For the collection of data, the JupyterLab environment on the Anaconda platform was used. With JupyterLab, a notebook file can be created allowing a mixture of prose and Python programming cells. JupyterLab was ideal for data collection as code could be broken down easily and data frames created could be easily inspected and manipulated at various sections in the data collection and cleaning process. JupyterLab, very similar to the popular Jupyter Notebook environment except it provides additional functionality, particularly provides a file tree which made navigating notebooks much easier. Both Jupyter Notebook and JupyterLab can therefore be used to run the *.ipynb* programs. The data collection system was split into two different notebooks:

- **000\_Petitions\_Data\_Collection.ipynb[1, 000]** – collection of petitions information
- **100\_Transcripts\_Data\_Collection.ipynb[1, 100]** – collection of other debate transcripts

## Programming Language

JupyterLab supports a number of different programming languages. However, Python was used in this data collection system as it supports many useful libraries for data retrieval and management.

## Petitions Information

The petitions information was collected in using the program *000\_Petitions\_Data\_Collection.ipynb*[1, 000]. This program was split up into the following parts:

- **001** - Load in CSV files downloaded from the petitions website [9]
- **002-004** - Append 3 additional columns to each data frame for petitions submitted during each of the three previous governments.
- **005** – Combine data frames to create one data frame containing all information about all petitions submitted on the petitions website [9].

### 001

Loading the CSV files as *pandas* data frames would provide the foundation for collection of petitions data.

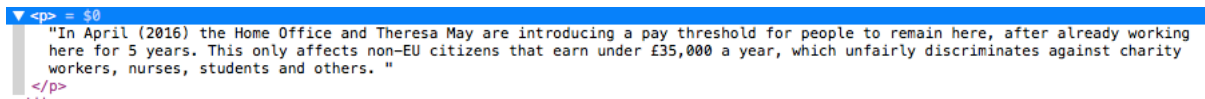
	Petition	URL	State	Signatures Count
0	Make British Sign Language part of the Nationa...	<a href="https://petition.parliament.uk/petitions/200000">https://petition.parliament.uk/petitions/200000</a>	closed	35203
1	Lobster Pots and Small Craft Safety – time to ...	<a href="https://petition.parliament.uk/petitions/200001">https://petition.parliament.uk/petitions/200001</a>	closed	10770
2	All high-rise tower blocks over 30 metres must...	<a href="https://petition.parliament.uk/petitions/200003">https://petition.parliament.uk/petitions/200003</a>	closed	271
3	Hold a referendum on the final Brexit deal	<a href="https://petition.parliament.uk/petitions/200004">https://petition.parliament.uk/petitions/200004</a>	closed	145119
4	Give all British citizens living abroad the ri...	<a href="https://petition.parliament.uk/petitions/200005">https://petition.parliament.uk/petitions/200005</a>	closed	10314

Figure 5 – Screenshot showing the first five rows of the CSV files as a *pandas* data frame

### 002-004

In order to collect the ‘information’ and ‘more-info’ for each petition I utilised the Python package *BeautifulSoup4*. *BeautifulSoup4* facilitates web scraping allowing data to be pulled from HTML files.

Within the source code of the webpage produced for each petition it became apparent that the fifth paragraph using the ‘<p>’ tag contained the text for the ‘Information’.



```
<p> = $0
    "In April (2016) the Home Office and Theresa May are introducing a pay threshold for people to remain here, after already working here for 5 years. This only affects non-EU citizens that earn under £35,000 a year, which unfairly discriminates against charity workers, nurses, students and others. "
</p>
```

Figure 6 – HTML source code taken from a petition page on the UK Petitions website [9] showing the information section of the petition.

In order to parse the webpage, the library BeautifulSoup4 was used. Creating a BeautifulSoup4(*soup*) object from a webpage enabled the use of the BeautifulSoup4 library. Therefore, the function *create\_soup(url)* was devised that returned the *soup* object for a given URL. I then defined the function *get\_info(url)* which would take a URL for a petition, create a soup object and use this to return the text from the fifth paragraph (‘p’ element) as a string.

If additional information was available under ‘*More details*’ then this information would be contained in the sixth paragraph using the <p> tag. However, this additional information was often not available as the author of the petition had simply not supplied more information. I defined another function *get\_more\_info(url)* which would, similar to *get\_info(url)*, take a URL for a petition, create a BeautifulSoup4 object and use this to return the text from the sixth paragraph as a string. To account for those petitions that were not provided with additional information, if the text in sixth paragraph contained “This petition is closed” that would mean that the petition did not include additional information and therefore the function would return ‘None’.

The third additional column, *transcript*, would contain the text data from the transcript of the debate if the petition had gone on to be debated in Parliament.

Initially, a function *find\_tscript(url)* would return the link of the transcript for the petition that had been debated in Parliament. If a petition had been debated in Parliament, then the petition webpage would include a link to the transcript of said debate. This link would have the label “Read the Transcript”. Therefore, the function *find\_tscript(url)* would search through all links on the webpage and return the link that contained the text ‘Read the transcript’. This is done by first creating a soup object using the *create\_soup(url)* method as before. The BeautifulSoup4 *find\_all()* method would then be used to find links with the ‘a’ tag that contain ‘Read the transcript’. The ‘a’ element represents a hyperlink in HTML. Figure 7 shows an example of the HTML source code to produce one such hyperlink. If there were no links found

containing this text, then the function would return `None` as this would mean the petition had not been debated.

```
<a rel="external" href="https://hansard.parliament.uk/commons/2019-04-01/debates/DAEA9200-D885-4370-B65C-2B82FF685AE9/LeavingTheEuropeanUnion">Read the transcript</a> =
```

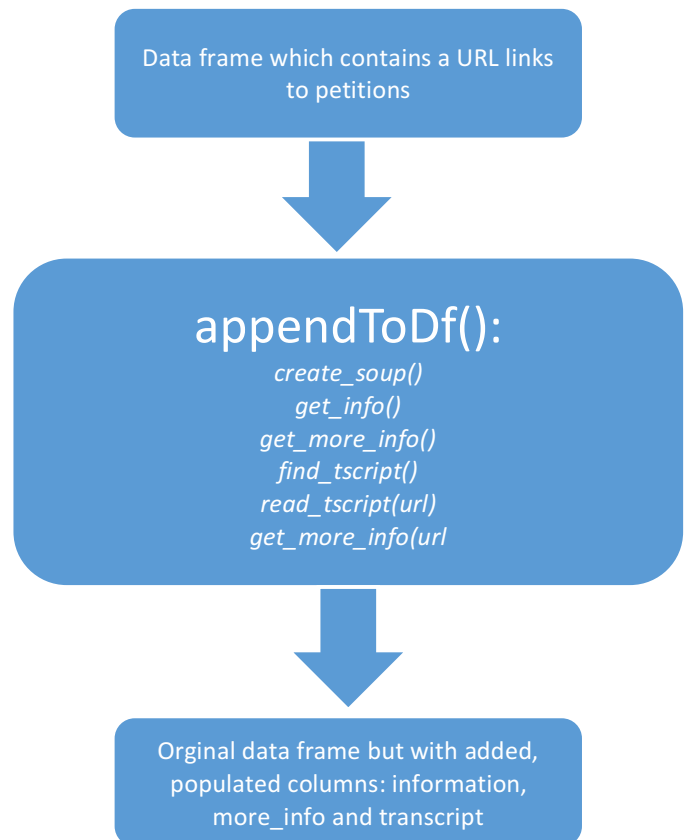
Figure 7- HTML source code taken from a petition page on the UK Petitions website [9] which produced the “Read the transcript” hyperlink.

A second function named `read_tscript(url)` would then go to the Hansard Parliament debates webpage containing the petition debate and return the string of the debate transcript if the petition had been debated or return `None` if not. In order to do this, again a `soup` object would be created from the URL but this time for the debate’s webpage. It became apparent that all text contained under the ‘p’ tag contained the words spoken in the debate as can be seen in Figure 8. Each paragraph contained as a ‘p’ element could then be stored in a list and joined together to produce one string containing every word spoken in a debate.

```
"It is a real pleasure to serve under your chairmanship, Mr Gray, and to lead this incredibly important debate on behalf of the Petitions Committee. As hon. Members will be aware, the Committee decided to schedule a single debate on all three Brexit-related petitions because we wanted to ensure that all three, having reached the 100,000 signature threshold, were debated as soon as possible, so that they would not be overtaken by events."
</p>
<p class="hs_Para">
  "It is entirely coincidental that the date is 1 April, but I must confess to hoping right up until noon that the Prime Minister was at some point going to reveal to the nation that the Government's entire handling of Brexit has actually been the most painstaking and elaborate April fool's day hoax in history, and that she does in fact have a plan to get us out of this mess. Regrettably, that did not happen, and we are still in a national crisis."
</p>
<p class="hs_Para">
  "Of course, as is now inevitable for anything related to Brexit, one of the e-petitions has already been overtaken by events: 29 March has been and gone and, three days later, we remain members of the European Union. As such, with just under two weeks to go until the next deal or no deal deadline, we find ourselves through the looking glass, debating potential Brexit outcomes here in Westminster Hall at exactly the same time as colleagues in the main Chamber deliberate the ways out of this ludicrous situation. A national conversation should clearly have been led and listened to by the Prime Minister right at the start of this historic process, rather than one being commenced against her will just before midnight on the Brexit clock."
</p>
```

Figure 8 – HTML source code taken from a debate on the Hansard Parliament website [10].

These methods for collecting the three extra sections of data, `get_info()`, `get_more_info()` and `read_tscript()`, for each petition would be combined in the function called `appendToDf(df)`. This function would take the original data frame with the 4 columns and append the three extra columns: `information`, `more_info` and `transcript`. This function was applied to the three data frames of petitions from the three different governments. The diagram beside outlines how the function `appendToDf` works:



005

Following this the three new data frames would be combined using the *pandas* function *concat()*. The resulting data frame would therefore contain all archived petitions and all current closed petitions along with the information for each. This data frame would then be saved as a *pickle* file which allows the data frame to be loaded in a different Python file in the same format as it was saved. This would make it easy to load in the data frame later in the study.

	Petition	URL	State	Signatures Count	information	more_info	transcript
0	resign	<a href="https://petition.parliament.uk/archived/petiti...">https://petition.parliament.uk/archived/petiti...</a>	closed	1849	We, the people of the United Kingdom of Great ...	None	None
1	Repeal the Digital Economy Act	<a href="https://petition.parliament.uk/archived/petiti...">https://petition.parliament.uk/archived/petiti...</a>	closed	1331	We the public are not satisfied with the proce...	Repeal the Digital Economy Act, and ensure any...	None
2	Lets leave the European Union	<a href="https://petition.parliament.uk/archived/petiti...">https://petition.parliament.uk/archived/petiti...</a>	closed	220	I would like the Government to leave the Europ...	None	None
3	Airbrushed Images in the Media	<a href="https://petition.parliament.uk/archived/petiti...">https://petition.parliament.uk/archived/petiti...</a>	closed	147	This petition calls on the Government to intro...	This petition believes this must be done to sc...	None
4	Reduce spending and taxes.	<a href="https://petition.parliament.uk/archived/petiti...">https://petition.parliament.uk/archived/petiti...</a>	closed	15	The house of commons should instruct the Treas...	Areas that should be considered include, but a...	None

Figure 9 – The first five rows of the final petitions information data frame.

## General Debate Data

As per design (Section 3.1), the general debate data collection process would be split into two sections:

- Collection of a list URL links to debates in the House of Commons.
- Scrape the webpage at each of these links to obtain a string containing all words spoken in each debates.

### Collection of a list URL links to debates in the House of Commons

BeautifulSoup4 facilitates web scraping of a webpage, however, an additional library Selenium is needed to crawl through a Javascript generated webpage which was necessary for collection of debates data. Selenium WebDriver can drive a browser as a user would, enabling Javascript generated information to become visible. In this project the Selenium WebDriver was used to generate the links for each of the 20 debates available on each page of the 'Find Debates' feature from the Hansard parliament website which could then be scraped using BeautifulSoup4 to obtain the URLs to each of these debates. The date range chosen for collection of debates was July 1<sup>st</sup> 2010 to December 31<sup>st</sup>

2018 which was entered into the 'Find Debates' feature. This would then give a URL particular to that date range which was '<https://hansard.parliament.uk/search/Debates?endDate=2018-12-31&house=Commons&startDate=2010-07-01&page=1>'. A loop would go through all the numbers from 1 to the number of pages available each time replacing the number at the end of the URL with the next one. In total the number of pages available was 1934 with, at most, 20 debates to a page which totalled to 38661 debates in the given date range. Within the loop all 20 URLs produced on each page would be appended to a list of URLs.

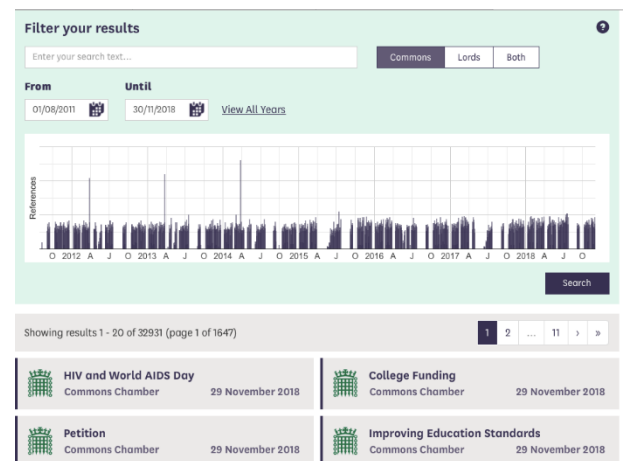


Figure 10 - Screenshot of Hansard website [9] debates feature.

**Scrape the webpage at each of these links to obtain a string containing all spoken words in each debate.**

As each of these URLs were links to Hansard debates, like the petitions data collection system, the `read_tscript(URL)` function could be reused to scrape the transcript of each of these debates. Each transcript was then stored as a row in a data frame of transcripts and saved as a pickle file as before which could be loaded in later.

	tscript
0	The Secretary of State was asked— 1. What pla...
1	Restoration And Renewal Of The Palace Of Westm...
2	11. What recent progress her Department has ma...
3	5. When he expects next to discuss with minis...
4	15. What plans he has for Sure Start children...

Figure 11- First five rows of the debates data frame

```
## Example of transcript
other_tscripts.tscript[0]

'The Secretary of State was asked— 1. What plans his Department h
as to attract more top science and mathematics graduates to be tea
chers. [801] There is clearly a problem with a shortage of special
ist teachers, as only 47% of mathematics teachers and 58% of combi
ned science teachers have first degrees in the subjects that they
teach. We need to do more to encourage pupils to study sciences an
d maths, and encourage graduates to enter teaching in those subjec
ts. Therefore, we are actively reviewing the routes into teaching
and bursaries, along with other incentives offered to well qualifi
ed people who want to teach science and maths. It is very good to
see my hon. Friend at the Government Dispatch Box. Will he ensure
that schools have sufficient powers and funds to offer generous re
tention bonuses to teachers of shortage subjects, and that schools
with retention problems are fully aware of such powers? I am very
grateful to my hon. Friend for those comments; it has been a long
time coming. We are certainly considering how schools can be furth
er encouraged to use the existing recruitment and retention pay fl
```

Figure 12 – Example an entry in the debates data frame

### 3.3 Testing and validating collected data

Both datasets would be very large so each entry could not be individually checked. A number of testing and validating methods were used as described below to ensure data was valid.

#### Inspection of data frames

As both data sets were stored as ‘pandas’ data frames, pandas library offers numerous methods for inspecting and testing the data collected. The most basic method for testing each data frame was to use pandas’ `head()` and `tail()` methods. These methods allowed for inspection of the first five rows and the last five rows of each data frame respectively and make sure data at the beginning and the end of the data frame was as expected.

#### Duplicate Entries

Another useful *pandas* method that was utilised was `describe()` which allows you to view how many of the entries into one column were unique. When describing the transcript column of the `other_tscripts` data frame it became apparent that many of entries were duplicates. The reason for this would have been that the `getLinks()` method for retrieving the debate links for each page in the ‘Find Debates’ feature would gather the links before the next page had loaded and therefore gather the same links twice. I included the method `sleep(2)`, from the *time* library,



inside the data collection program which would allow the Selenium WebDriver time to load the new page before the new debates link were collected. This resulted in significantly fewer duplicate links being returned but a number of duplicates still remained. Therefore, for the returned list of links only unique links would be iterated through.

### Minimum length validation

I did minimum string tests when collecting both datasets to ensure all entries were as expected and did not include empty or very short text entries.

The *information* column of petitions data would contain the information about the petition and would be used for creating the petitions LDA model (Section 4). Therefore, the *information* column would need to be at least a sentence long. I removed all rows with an *information* column with a string length of less than 20.

It became apparent when inspecting the lengths of the strings returned from the petitions data collection that many of these strings were short. From viewing the links that resulted in these short strings, it became apparent that these transcripts simply represented a statement, by the Speaker of the House or other, and not a debate, as can be seen in Figure 13. To ensure that all transcripts in the data frames were of full debates and not just statements, debates with a string length of under 1000 characters were removed.



Figure 13 – Screenshot taken from the Hansard Parliament debates website [10] showing a statement rather than a full debate.

### Petition Data – Signature Count

When inspecting the petitions data, it became apparent that many of the petitions received very few signatures and would therefore not be representative of public opinion. To alleviate this problem only petitions that received over 20 signatures would be considered.

### Dealing with dead links

It became apparent during the collection of both datasets that a small number of URLs that were expected to address either a petition or debate were dead links. This would cause the data collection program to crash. To deal with this, the two data collection systems would simply return *None* if the desired petition/debate could not be found. Subsequently data frame rows that did not return a petition/debate would be removed.

### Final data set

## 1. Petitions

**Title-** title of the petition given by the author

**URL** – link to the petition on the UK petitions website

**Signatures Count** – number of signatures at time of data collection

**information** – details of the petition provided by the author

**more\_info-** additional details provided by the author

**transcript** – the transcript of the debate

## 2. Transcripts

**tscrip**t- transcript of a debate in parliament

Number of documents

- |  |        |
|--|--------|
| 1. Petitions – <i>petitions_info.pickle</i>  | 24,442 |
| 2. Debates – <i>other_transcripts.pickle</i> | 20,961 |

## 4. Topic Modelling

Topic modelling is an approach to natural language processing that involves the unsupervised training of a probabilistic model in order to infer abstract topics that occur in a set of documents (corpus)[11]. Latent Dirichlet Allocation (LDA) is the most common method for producing topic models.

### 4.1 LDA

LDA is a machine learning method developed by David Blei, Andrew Ng and Michael I. Jordan, first introduced in 2002. A LDA model is probabilistic model for collections of discrete data. A common application of LDA is in modelling a group of text corpora(documents). Modelling a group of documents using LDA will deduce a number of latent patterns (*topics*) that define the corpus. Each topic is simply a group of words with the probability distribution that each of words would appear in the topic. Each document is then characterised by the probability distribution of the document for each topic. [11]

For this study the following two LDA models were developed based on the following text data:

- **Petitions details-** concatenation of *information* and *more\_info* columns from *petitions\_info.pickle*
- **General debates transcripts** – *other\_tscripts.pickle*

The topic distributions from each model could then be compared and contrasted.

### 4.2 Implementation

The two data frames *petitions\_info.pickle* and *other\_tscripts.pickle* collected in the programs *000\_Petitions\_Data\_Collection.ipynb* [1, 000] and *100\_Transcripts\_Data\_Collection.ipynb* [1, 100] would be loaded in a new notebook file named *200\_Model\_Creation.ipynb*[1, 200]. as a *pandas* data frame.

#### Text Data Used

The concatenation of the *information* and *more\_info* column in each row of the *petitions\_info* data frame would provide the text data for the first LDA model (*petitions\_ldamodel*). Each row *other\_tscripts* data frame would simply provide the text data second LDA model (*lda2*).

## Pre-processing

Data was initially cleaned and validated in the data collection process through methods of inspection, duplicate detection and minimum length validation (Section 3.3). Data would require further pre-processing prior to modelling in order to ensure the training process was as efficient as possible. Due to the large volume of data, particularly debates transcripts, it was necessary to speed up the training. The method *prepare\_for\_LDA()* was defined that would employ the following pre-processing methods:

- **Removal of stop-words**

Stop-words are commonly used words (such as “a”, “and”, “the”) that offer little to no additional information about a text. These words would therefore need to be removed to reduce processing time and ensure topics distributions did not include these words. The Python library NLTK (Natural Language Toolkit) facilitates this by providing a list of these stop words which can then be removed from all texts.

- **Lemmatization**

Lemmatization involves converting forms of a word into its root word that they can be analysed as one element. Words with the similar context such as “walked” and “walks” will be treated as two occurrences of the lemma “walk”. This would again cut down processing time and avoid the word distribution of a topic from the LDA model including many forms of the same root word with unreliable weightings. NLTK provides a network of relationships between words. NLTKs *WordNetLemmatizer()* method would then be used to return the lemma for each word.

- **Tokenisation**

The string of text data would need to be split up into a list of only words. The process of splitting a text up into tokens is known as tokenisation. The library *Spacy* facilitates tokenisation by removing spaces and punctuation marks and returning a list of words.

## Training the models

*Gensim* is open source Python library which provides functionality for unsupervised topic modelling techniques. As previously discussed, the technique used in this project would use LDA. *Gensim* provides two methods for training *LdaModel()* and *LdaMulticore()*. *LdaMulticore()* can be up to three times faster the older *LdaModel()* as it uses all CPU cores

to parallelize model training. Therefore, the method *LdaMulticore()* was used for training both models which takes the following parameters:

- *corpus* – structured set of documents with every word in a document represented in number form. Each number represents an ID which is referenced by a dictionary
- *id2word* – this is the dictionary for converting each ID in the corpus to a word
- *num\_topics* – number of topics desired

The number of topics required from the LDA for both models was decided to be 20 topics for ease of visualisation.

*gensim* offers functionality for converting the tokenised list of words for each document into the corpus and dictionary format required for the method *LdaMulticore()*. The first method

*corpora.Dictionary()* takes the text from every document provided, assigns every word an ID in a dictionary and returns said dictionary. The second method *doc2bow()* would then create a corpus of documents with every word in a document represented in number form with the number referencing the position of the word within the dictionary. This process of converting a document into number form is outlined in Figure 14.

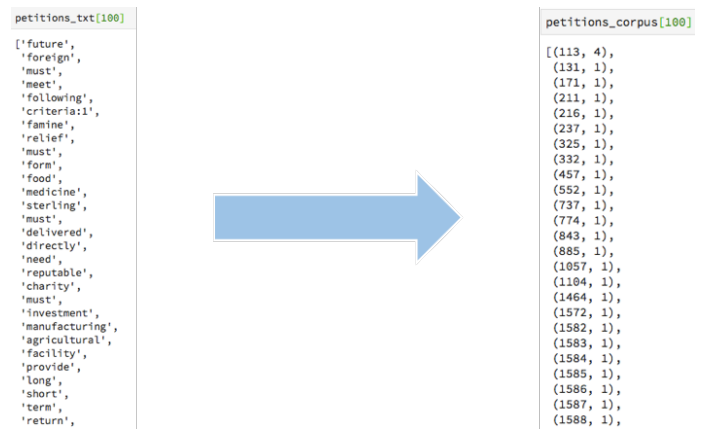


Figure 14 - Outline of the process of converting the text data into numeric form.

As the topic distribution of the two models will be compared, a universal dictionary must be created. This dictionary was created combining the two lists of tokens created from the two sets of documents.

The *LdaMulticore()* method is then performed on the two sets of tokenised documents to produce two LDA models.

### 4.3 Analysis of Models

The two sets of documents are now modelled as two different LDA models. Each of these models have discovered the 20 latent topics that describe the whole set of documents. Each document can then be described as the probability distribution of those 20 topics. This study aims to compare the two sets of documents by comparing the difference between these two models. Each topic is characterised by the likelihood of a word appearing in that topic. Analysis of the models was carried out in the notebook *300\_Analysis.ipynb*[1,300].

### Similarity Measures

In order to compare the two models, we would need to compare the 20 topics in one LDA with each of the 20 topics in the other. Each topic model is a probability distribution of the words in that topic. Therefore, distribution similarity measures such as *Cosine Similarity* and *Kullback-Leibler Divergence* were considered. The problem with these distribution measures was when it came to words not featuring at all in some topics which would skew the results.

### Jaccard index

The distribution similarity measure used to compare topics between the two models in this project would be The Jaccard index [12]. The Jaccard index is defined as the size of the intersection divided by the size of the union of the two data sets and is given by the following equation:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Figure 15 – Equation describing Jaccard's Index

With reference to this project, the *n* most relevant words in each topic are compared with the *n* most relevant words in another topic. The *gensim* library provides the function *lda.diff()* which will calculate the difference in topic distributions between two models. In this project the *lda.diff()* function takes the following parameters:

- **other** – the second LDA model to be compared (*lda2*)
- **distance** – the distance metric to calculate the difference in distributions with. As mentioned before we will be using The Jaccard index as a similarity measure. In this case the Jaccard Distance between the distributions is being measured which is a measure of dissimilarity and simply 1-(Jaccard Index).

- **num\_words** – This is the number,  $n$ , of the most relevant words in each topic that would be considered when finding the Jaccard Distance. After inspecting the distributions of words in each topic the decision was taken to use  $n=100$  as the majority of words which were less relevant than the top 20 had very small probabilities ( $p<0.001$ ) of being in the topic and therefore were not considered.

The output of the `lda.diff()` function is a difference matrix with each element representing the distance between one topic and another. As the distance between the distributions represents the dissimilarity, each element in the matrix is subtracted from 1 to give the similarity matrix. This similarity matrix was visualised as a heat map in Figure 16.

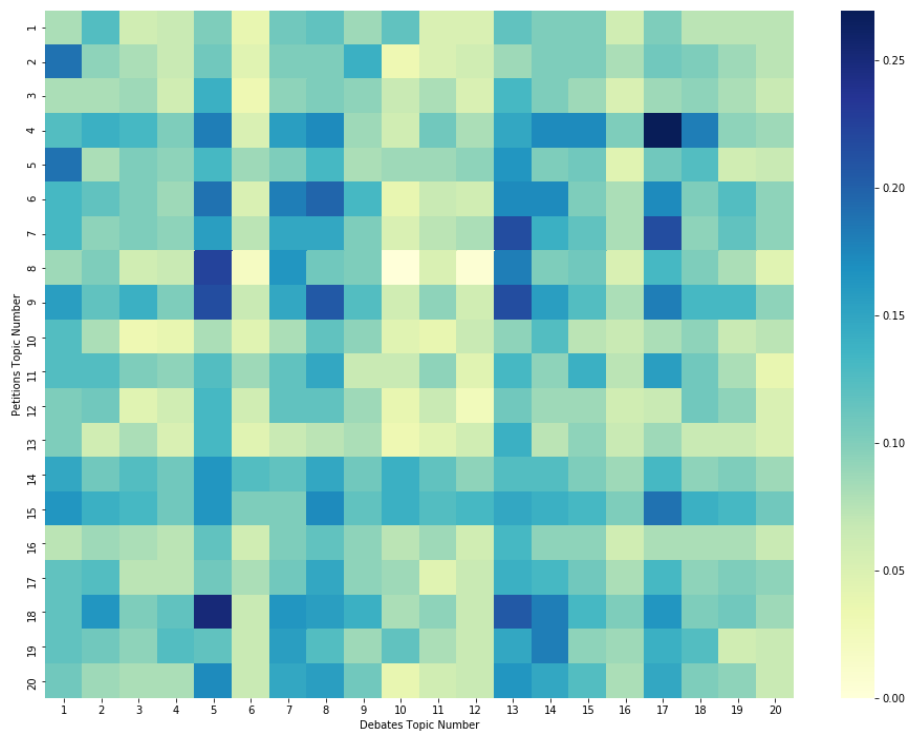


Figure 17 – Heat map showing topic similarity scores for each topic in each model

From the heat map (Figure 16) one can easily view which petitions topics are strongly correlated with a topic in the debates models. For example, petitions debates topic 4 is very similar to debates topic 17. This single topic to topic comparison, however, does not give an over all picture about how well a petition topic has been debated. In an attempt to achieve this a line graph was plotted which displayed the similarity matrix with each line representing each petition topic plotted against its similarity with a debates topic Figure 17.

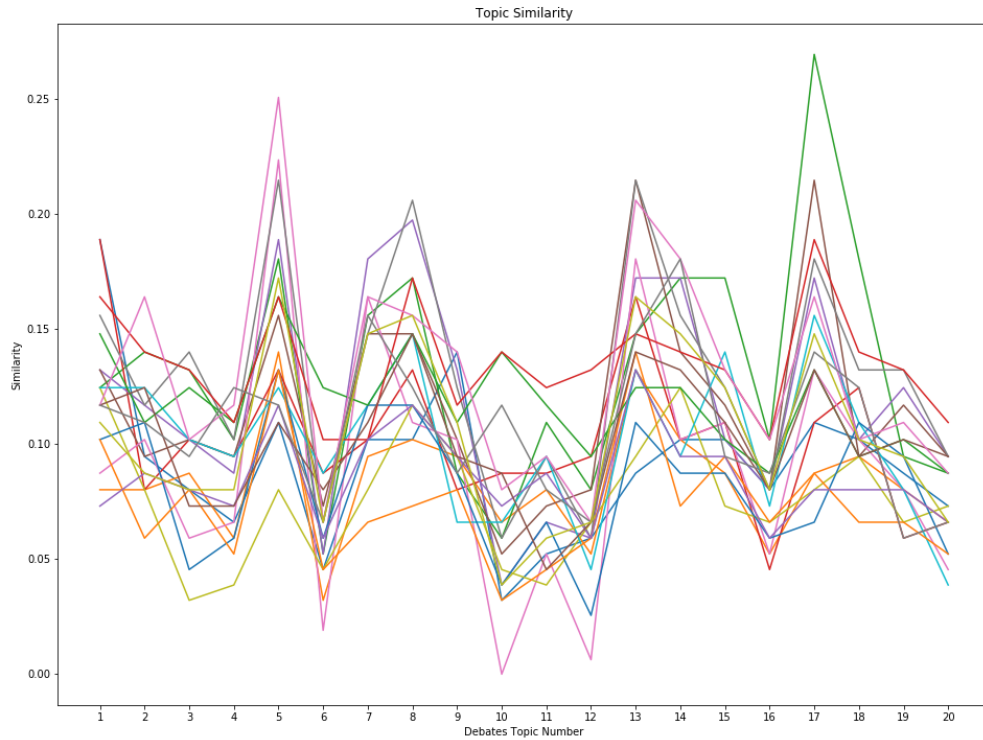


Figure 17 -

From the line graph one can deduce which debates topics tend to score higher in similarity when compared with the petitions topics. Spikes at Debate Topic numbers 5, 8, 13 and 18 indicates that these topics are generally more similar to Petitions topics and therefore it is these debate topics that are more likely to feature in petitions. However, again this does not fully provide the answer to which petitions topics are more likely to be debated in Parliament. To do so a tool was developed (Section 5) using this graph as a starting point that would enable the user to select a petition topic and see how similar this topic is to all the other debate topics.



## 5. Topic Explorer

### 5.1 Design

An application would be developed that would enable a user to explore the information gained from this project. Information developed from the previous sections would be visualised so that deductions could be made about the contrast in topics discussed in petitions and those discussed in debates.

#### User Interface Design

Many designs were considered for the user interface for the topic explorer application. An initial design for a user interface was discussed during a supervisory meeting (Figure 18). This design shows the lists of topics for each of the two LDA models. It was proposed that the similarity between topics in both the petitions and debates could be compared in a separate window along with the similarity scores.

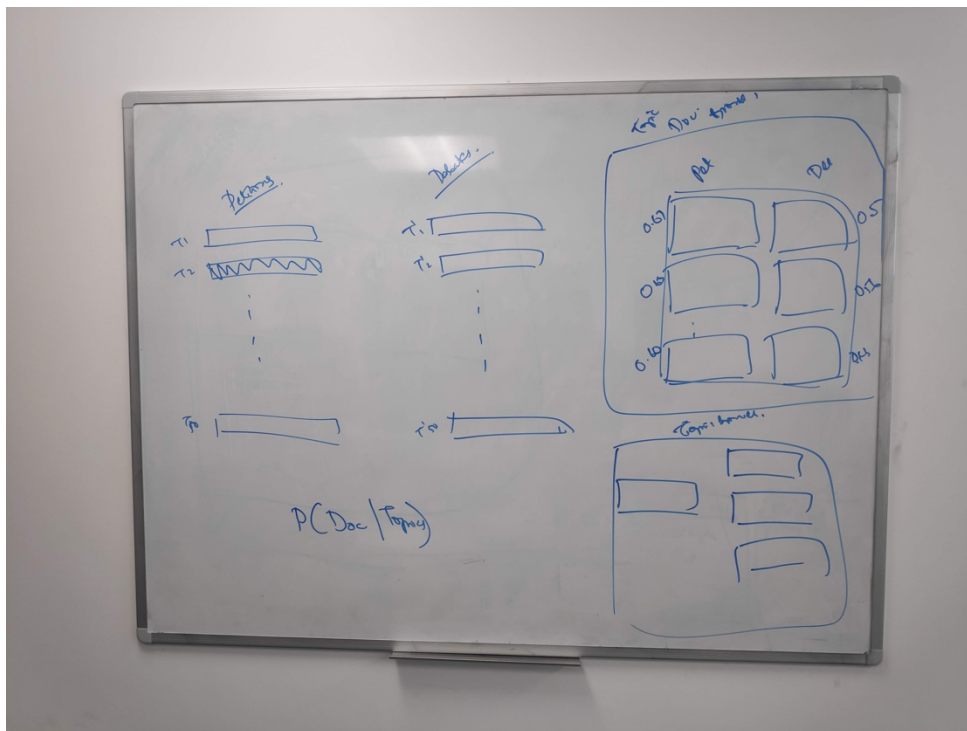


Figure 18 – Initial design conceptualised in supervisory meeting.

The final design would include the following two windows:

- **Main Window**
- **Topic Window**

## Main Window

It was later decided that an alternative to showing topic similarity as numbers would be to portray the similarity results graphically. The functionality and layout of the final design would be as described below and in Figure 19:

- On the left-hand side of the window all of the petitions topics would be listed and on the right hand side each of the debates topics.
- Each of the topics are simply displayed as the top 5 most relevant words in the topic.
- A graph would be placed in the centre of the window.
- Each of the petition topics would be presented as a button that, when clicked, would place a tick beside the topic.
- *Clicking* the topic again would remove the tick.
- All topics which had been ticked would then be drawn onto a graph in the middle of the interface.
- The line graph would plot each topic as a line in the graph representing its similarity with each of the 20 debates topics. The similarity matrix produced in `300_Analysis.ipynb[1,300]` would provide the data for said plot.
- Clicking on the title for each petition topic would open the topic window for said topic.

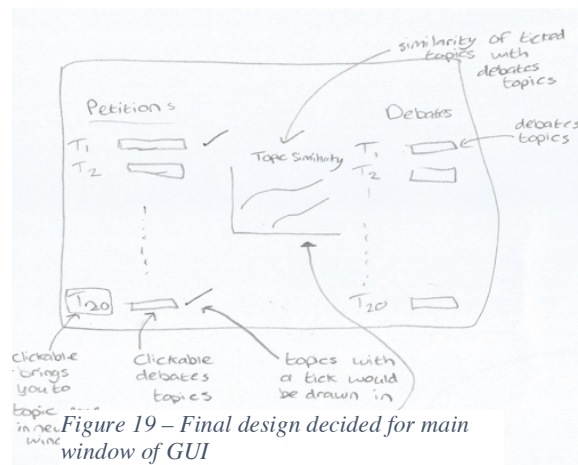


Figure 19 – Final design decided for main window of GUI

## Topic Window

Viewing the topics as a small list of words would be abstract and may not be that easy to understand. In order to supply context to each of the topics another window would be added to the system. This topic window would be opened when a topic title was clicked. This window would show the top 5 petitions relating to each topic. When one of these petitions is clicked

the webpage for that petition would open in a browser. The top 5 petitions for each topic had been discovered in Section 3 in the file *300\_Analysis.ipynb [1,300]* and would simply be loaded in on initialisation of the program as a data frame which could be indexed when necessary.

## 5.2 Implementation

### Development Environment - PyCharm

JupyterLab was used as the software development environment for data collection (Section 3) and for modelling data (Section 4) as it provided functionality for inspecting and manipulating data frames, visualisation of data and code could be easily broken down for troubleshooting. PyCharm offers much more features for development of a full application and would therefore be more suitable for implementation of the Topic Explorer GUI.

Data frames that had been produced from the running of the Jupyter Notebooks from 000-300 could still be easily used and employed in the application. Often methods, such as graph creation, were developed and tested in JupyterLab and then employed in the application. These methods can be seen in the file *003\_Analysis.ipynb[1,300]*. The program used for the implementation of the Topic Explorer GUI was *400\_Topic\_Explorer.py[1,300]*.

### Loading in the data

This program first involves loading in information which had been gathered and developed in the previous files. The following files would be loaded in at the beginning of the program:

- The data frame containing the five most representative documents for each topic (developed in *300\_Analysis.ipynb[1,300]*). This would be used to give context to each of the latent topics by providing the documents that most represent each topic. todo
- The data frame containing the topic similarity between the two models (developed in Section 4 using *300\_Analysis.ipynb[1,300]*). This data frame would be used to create the graph the interactive topic similarity graph.

## GUI Implementation

The implementation of the GUI was achieved by utilising Python's inbuilt library *Tkinter*. *Tkinter* provides functionality for creating a user interface application along with many useful methods for organising widgets within a user interface. A user manual was created to explain how to use the program and is available in the project GitLab repository [1].

### Main Window

The main window would open on initialisation of the *Tkinter* object. The following table gives general description of how key features were implemented in the main window as described in the design (Section 5.1):

Feature	Implementation
Layout	A <i>Tkinter Frame</i> widget was created from the main window. The <i>Frame</i> object provides a rectangular containment for other widgets. Other widgets such as <i>Buttons</i> and <i>Labels</i> can then be placed at co-ordinates within the existing <i>Frame</i> allowing the creation of an organised and visually appealing GUI.
Displaying Debates Topics	Each of the 20 topics would be displayed as Labels on the right-hand screen of the window.
Displaying Petitions Topics	<p>Down the left-hand side of the screen would be two buttons for each of the 20 debates topics.</p> <ul style="list-style-type: none"> <li>One button labelled with the text 'Topic n' with n being the topic number. This button would open a new window <b>Petitions Window</b>.</li> <li>A second button containing the text of the top 5 words for each topic which was accessed through the data frame <i>petitions_topics_sorteddf.pickle</i> produced in <i>003_Analysis.ipynb[1,300]</i>. When this button was clicked a tick was placed in the column beside. If there was already a tick in the column beside it this tick was removed. In order to implement this a list would store each Labels in the next column beside the topic. If the Label didn't already contain a tick a tick would be added as that Label and if it did contain a tick the tick would be removed.</li> </ul>

Feature	Implementation
Plotting Graph	<p>The method for producing the graph involved utilising the library <i>Matplotlib</i>. This library provides functionality for creating graphs and figures.</p> <ul style="list-style-type: none"> <li>JupyterLab was used in the development and testing of this graph which can be seen in <i>003_Analysis.ipynb[1,300]</i>.</li> <li>In order to display the graph inside the <i>Frame</i>, a <i>Canvas</i> would be displayed in the middle of the window.</li> <li>The graph produced in through <i>Matplotlib</i> would then be drawn onto said <i>Canvas</i>.</li> <li>The data for the graph would be indexed from the topic similarity data frame produced in <i>003_Analysis.ipynb [1,300]</i>.</li> <li>A plot of topic similarity between ticked petitions topics and all debate topics would be plotted.</li> </ul>

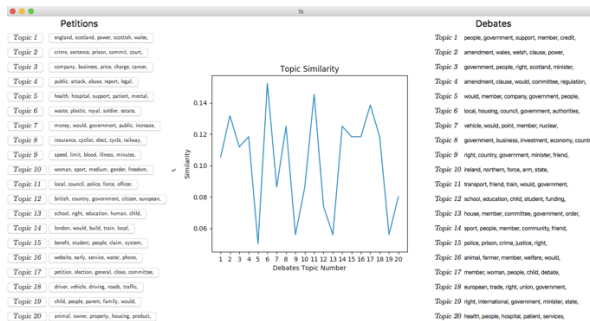


Figure 20 - Main Window showing Topic Explorer program one topic selected.



Figure 21 - Main Window showing Topic Explorer program with multiple topics selected

## Topic Window

The topic window would be opened when clicking on the title for a topic eg. “Topic 1”. This topic window simply states the titles of the most relevant documents for each of topic. A data frame was developed producing the five most relevant documents for each data frame was devised in the program `300_Analysis.ipynb[1,300]`. This data frame would then be indexed to display the titles of five petitions that best represent the topic. Each petition would be in the form of a Button which when clicked, would open a web browser to the link containing that petition.



Figure 22 - Example of the Topic Window for Petitions Topic 9

## 6. Conclusion

### 6.1 Project review

Over the course of this project, a wide scope of computer science methodologies had been utilised including Data Mining, Topic Modelling, Data Analysis and GUI Development.

The Data Mining techniques of web scraping and web crawling were used for collecting data from the internet and developing the two data frames of petitions and debates information (Section 3).

The text data was subsequently validated (Section 3) and then pre-processed (Section 4) using various techniques such as duplicate detection, remove of stop words and lemmatisation.

Following this the two topic models were trained. One model was training on the details provided about petitions collected from the UK Petitions website [9] and the other was trained on the UK Parliament debates collected over the same time period collected from the Hansard Parliament website [10].

Finally, both models were analysed (Section 4) and a Topic Explorer GUI was developed that would aid in analysis of topics produced from the topic modelling.

Overall, I believe this project is a stepping stone for answering the question about how petitions influence debates in Parliament by identifying which petitions topics have been sufficiently debated in Parliament and which have not.

### 6.2 Limitations and Possible Improvements

#### Search feature

#### Petitions Data Signature Count Weighting

As long as a petition received over 20 signatures it was given equal weighting in the training of the petitions LDA model. Example: a petition with 30 signatures and one with 500,000 signatures would both provide the text data for a single document in the model. For this study

a general approach was taken whereas more systematic approach to weighting the documents may have provided a more reliable insight into public opinion.

### **Tuning LDA models**

Designing an LDA model requires the stating of a desired number of latent topics for the model to find. It was decided to include twenty topics for the reason that it would be a coherent and understandable number for the user. Two metrics for scoring the quality of an LDA model are coherence score and log-likelihood. A coherence score can be calculated for each model and is used for testing the quality of learned latent topics. A higher score generally means topics are better distributed and give a better representation of a corpus of texts. Using the number of topics that resulted in a higher score, using one of the aforementioned tuning metrics, would provide an optimal model. This proved difficult to implement as the length of time taken to run each model would be too substantive for this project.

### **Topic Similarity Measure**

Jaccard's distance was used as a similarity measure for the topic distributions. A notable downfall to this method is that each of the  $n$  most relevant words are equally weighted and magnitude of a word's probability is not considered. As long as its probability is high enough to put it in the top  $n$  most relevant words.

## **6.3 Further Studies**

The Topic Explorer GUI developed (Section 5) can provide an insight into which petition topics are discussed in Parliament and which are not. Further studies may explore the reasoning behind the disparity between the topics which are discussed in the UK Parliament and those which are neglected, despite calls from the public to discuss this range of topics on the UK petitions website.



## References

1. Mallon, D. *Project GitLab repository* Available:  
<https://gitlab.eecs.qub.ac.uk/40154387/CSC3002>
2. Hough, R., 2012. Do legislative petitions systems enhance the relationship between parliament and citizen?. *The Journal of Legislative Studies*, 18(3-4), pp.479-495
3. Department for Digital, Culture, Media & Sport.(2011, Aug. 11). *E-petitions*[Online]. Available: <https://www.gov.uk/government/news/e-petitions>
4. BBC News. (2015, Oct. 30). *Are e-petitions a waste of time?* [Online]. Available: <https://www.bbc.co.uk/news/uk-politics-34476264>
5. Leston-Bandeira, C., 2002. Parliament and citizens in Portugal: still looking for links. *Parliaments and citizens in Western Europe*, pp.128-52
6. Bogdanor, V., 2009. *The new British constitution*. Bloomsbury Publishing.
7. Carman, C. 2006 *The Assessment of the Scottish Parliament's public petitions system, 1999-2006*. Scottish Parliament.  
<https://archive.parliament.scot/business/committees/petitions/reports-06/pur06-pps-assessment-01.htm> accessed Jan 23rd 2019-04-27
8. A. Tait. (2017, Jan. 30). *Do online petitions actually work? The numbers reveal the truth*[Online]. Available: <https://www.newstatesman.com/science-tech/2017/01/do-online-petitions-actually-work-numbers-reveal-truth>
9. UK Government. *Petitions: UK Government and Parliament* [Online]. Available: <https://petition.parliament.uk>
10. UK Government. *Hansard: The official report of all parliamentary debates* [Online]. Available: <https://hansard.parliament.uk>
11. Blei, D.M., Ng, A.Y. and Jordan, M.I., 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan), pp.993-1022.
12. Jaccard, P. (1901) Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles* 37, 241-272.