Relationship Identification from Text

Knowledge Discovery & Management Project 2016/17

Edward Fleri Soler
247696M
Faculty of ICT, University of Malta
Msida, Malta
edward.fleri.14@um.edu.mt

Jenny Attard
16696G
Faculty of ICT, University of Malta
Msida, Malta
jenny.n.attard.14@um.edu.mt

ABSTRACT

With the introduction of the internet and the shift towards digital media, large data repositories have been sprouting, storing terabytes of data and millions of articles. Digital encyclopaedias such as *Wikipedia* host a voluminous number of factual articles about anything and everything, ranging from public figures and animal species, all the way to mathematical models and historical events. This fact base has proven highly beneficial for scholars, students and every day Joes alike. However, lengthy textual articles are often hard to fully absorb, and hold highly useful information just beneath their skin.

This is where the application of knowledge discovery and handling techniques come into play. In this paper, we shall explore the design, development and application of a variety of techniques to extract, analyse and report useful findings from *Wikipedia* articles. The main focus of this system shall be to produce graphs visualising the underlying relationships between prominent Maltese politicians and other entities they are involved with. This will provide a summarised glimpse into the lifestyles, motives and partnerships of these specific individuals. The application of moralization and community detection algorithms will enable us to outline communities within the relational graphs and observe the entities which most prominently interact with one another.

The implementation shall start by querying *Wikipedia*'s API, *Wikimedia*, before applying natural language processing tools to analyse and process textual data. Relation extraction together with co-reference resolution and named entity recognition shall then be applied for the extraction of meaningful relations between the specified entities. These relations will then be visualised, following the application of moralisation and the *Bron-Kerbosch* (community detection) algorithm.

General Terms

Algorithms & Languages

Keywords

Relation Extraction, Co-reference Resolution, Named Entity Recognition, Moralisation & Community Detection

1. INTRODUCTION

From the creation of the interconnected web, in the late 20th century, abundance of information was available for everyone with a simple touch of a button. According to *Cisco*'s Visual Networking Index Initiative, by the end of 2016, the internet was expected to reach over 1 zettabyte of data transfer per year and doubling by the year 2020 [1]. Moreover, statistics show that in total, 40,000 *Google* searches were made every second [2] and 300 hours of video was uploaded to *YouTube* every minute in the year 2016 [3].

With this rapid growth of information on the web, efficient search algorithms are very important to help analyse this large amount of data, both on the web in general and also across the 1 billion plus webpages which are currently available on the internet [4]. This form of analysis allows for the understanding of underlying relations and connections between people, places and entities in general.

From this freely accessible data, new information could be found and learned by analysing relations, communities or groups of data within an article or a source. This is where our assignment comes into place. A number of algorithms will be used to traverse *Wikipedia* pages and get the most information out of specific articles. With such a call, further processing could be implemented to display key findings from this data within a visual graph, allowing for ease of observation and understanding of relations, communities and other underlying features.

Moreover, the use of graph diagrams will be used as findings show the people learn more by using this approach rather than reading the same content in a textual form [5]. Finally, moralisation and community algorithms will also be applied to clearly show if any relationships and more important information exists within related or nearby entities.

2. OVERVIEW OF RELEVANT BACKGROUND RESEARCH

Graphs are used in many different areas such as in computer science, biology and also chemistry. These visualisations are used to represent a difficult structure or to represent a molecular map [6] but although they represent different topics they are still used to solve the same problem. This problem is to support a better understanding when conversion from textual to visual structure is made.

Moreover graphs are also used to identify relationships between different sources. Taking popular networks into account such as *Wikipedia* or *Facebook*, graphs can be used to analyse connections between the different articles or group of friends in the respective web pages. This type of analysis is very important especially in web pages making use of a lot of data. *Wikipedia* alone consists of millions of articles and *Facebook* has more than 1 billion users registered on its network, therefore calling to efficient the need for analysis on complex data so as to identify any relevant information.

[7]

Additionally, analysis on *Wikipedia*, for example, may reveal connections between different politicians, both between members within the same party and also across the different parties. It may also reveal similar places that people attended, such as previous schools or events. Keeping the example of politics, analysis can also display stronger bonds between members within the same party as well as within opposing parties.

Graph diagrams began in the 1960s, with a common method used to show such diagrams being the use of *Node-Link* [6,7]. Here a graph is represented by vertices referring to nodes in the network and straight or curved edges between nodes to represent the relation that a node has to any other.

Since edges will be drawn to connect the different nodes, a problem may be encountered when a lot of entities are available in the graph and therefore it might be harder to read or understand the visual diagram. Such a case is highly probable in the real world and this algorithm alone might not be that adequate. In order to solve such a problem [6] makes use of a different layout structure. Space-filling techniques are adopted to visualise a sufficiently large graph by displaying the relationship between the nodes implicitly. An example of a Space-filling technique is *Treemaping* but although it may solve the problem of clustered data it lacks in the ability to understand the subject at hand.

Moreover, in the case of a dynamic graph, the graph may change over a period of time where a node addition or deletion may disrupt the whole graph. With this issue, connections to other nodes with or without the current entity should be made appropriately. Several solutions come into place with this concern, one solution is to display nodes and show the progression of a dynamic graph using time-steps [7]. Following this approach, the graph gets bigger and more informative as time progresses.

Another problem that a substantially large graph can cause is the inability to read it. It would therefore be ideal to allow the user to zoom in and out according to what is needed without the loss of any details. This is also a problem mentioned in [6] and it was found that *Fish-eye* methods can be used to integrate with the graph without losing any information or clarity of the visualisation diagram itself.

Furthermore, when working with a lot of information, the graph should show the relevant and most important data to clearly show what it represents. On the other hand, animations such as displaying more nodes while the graph progresses should be kept at a minimum as it can be overwhelming in certain cases. Otherwise, sub-graphs could be cropped from the main graph to be displayed in better detail and reduce the confusion that multiple connecting edges bring about. [7]

Sub-graphs can be obtained by using community detection algorithms which offer the possibility to discover *"organizational principles in networks"* [8] by grouping nodes having some overlap with other nodes together. This grouping can only be performed if a specific node is connected to any other node in its community. Methods proposed so far to identify community networks in large and complex graphs are not suitable because of their computational cost [9]. Moreover, community detection usually focuses more on the structure of a graph rather than clustering which focuses more on node attributes [8].

One algorithm which is used to identify any community detections in a graph is the *Girvan Newman* algorithm. This works by calculating the betweenness of all the edges in the graph, with the relational edge having the highest betweenness value being deduced. Since all nodes and edges are connected to each other it implies that they are dependent on both the edges as well as the nodes. Addition or deletion of a node/edge may affect the graph structure. For this reason, recalculation of those nodes and edges which were effected by such an action is implemented. This process is repeated a number of times until no edge is evident.

The *Girvan Newman* was found to have 2 disadvantages: firstly, all nodes are expected to have the same degree and secondly, all its communities should also have the same size [10].

Another algorithm also used to detect communities within a network is the *Bron-Kerbosch* algorithm which detects maximal cliques within a graph in order to detect communities. A problem with the *Bron-Kerbosch* is that when this algorithm is applied to larger graphs it might take a significant amount of time to complete successfully. [11] tested this algorithm using the classical *Bron-Kerbosch* approach and also with some modifications on this algorithm to represent a different performance and tested on graphs having different number of parameters i.e. different number of nodes and edges. Findings showed that the algorithms managed to execute upon graphs having up to 1000 nodes and 30% chance of connection between nodes. Graphs larger and more complex than this were manually interrupted due to the significant time it took to execute. Interruption was made after 20 to 30 minutes of execution. The execution time also depends on the programming language used, for instance in this case Java was used to implement the algorithms and since java runs on a Virtual Machine it takes longer to execute a snippet of code compared to other programming languages [11].

Computational efficiency is essential, with algorithms requiring minutes to compute being considered obsolete or highly unappealing. In order to speed up processing, linear algorithms with a run time capped at O(nlog2 n) are considered suitable. This is what the authors of [9] propose where 1 application of their algorithm is used to analyse sale items on a large online retailer's web page, where links between items are made if the item is purchased by a client frequently.

A greedy optimisation for their algorithm is used to place each node, save itself, in the structured network to represent a community. The 2 communities consisting of just 1 node are combined if such a combination results in the highest value of the probability of the relational edge existing between the nodes. The algorithm finishes executing once a single community is evident, this is when *n-1* connections are made in a graph having *n* vertices. [9]

Since these detection algorithms identify strong *"intra-connected modules"* which usually refer to important information, *"map equations"* are usually incorporated with other algorithms to find, evaluate and display modularity in graphs. *"Map equations"* are flexible and can be used to identify multi-levels in directed networks. [12]

3. DESCRIPTION OF METHODS USED

This relation identification system follows a logical workflow, as do most systems of similar functionality. Beginning with the acquisition and organisation of raw data in the form of text, the system follows with the processing of this

raw data and the extraction of meaningful relations. These relations undergo further processing and fine tuning before being visualised through a graph visualisation library. The programming language of choice for this system was decided to be Java, based on its functionality with API requests, as well as the numerous natural language processing libraries available for the extraction of relations.

The source of relation data for this system has been defined to be the *Wikipedia* digital encyclopaedia repository. This repository consists of over 5 million articles in the English language alone, from a range of topics and entities. The entities of interest have been confined to 8 prominent Maltese politicians, each with their own informative *Wikipedia* article.

3.1 Data Requests

The first step in the extraction of relations between these entities involves the request for raw data on each of these entities. Specifically, 4 requests are made to *Wikipedia* by means of 4 different Java functions: *getRawText, getLinks, getLinksHere & getInfoBoxRels*. These functions are invoked to get full page data, outgoing page links, incoming page links and *Wikipedia* information box data respectively.

All functions follow a similar pattern, beginning by submitting a GET request to *Wikipedia* through the implementation of the *Wikimedia* API. This request dictates the type of data requested, pertaining to which entity and in which form for it to be delivered. The full page data returned following the request is stored in a text file for purposes of efficiency, removing the need to re-download these text files on each consecutive program execution. Meanwhile, the incoming and outgoing links, together with the info-box data are stored in variables and used immediately.

3.2 Info-Box Relation Extraction

Basic pre-processing is performed on the full page data, removing unimportant text and symbols from the file. Meanwhile, data returned by the info-box request undergoes full processing for relation extraction within the *getInfoBoxRels* function. The *Wikipedia* information box provides a summary of its respective entity, listing important data. In the case of politicians, some of the listed information includes term start and end dates, birth place, educational institutions attended and predecessors/successors.

This function begins by marking and extracting salient data pertaining to relations from the info-box query response. A relationship linking the current main entity (example *Joseph Muscat*) to another entity through a label is considered. For example, considering Figure 1 on the right, a relation between *Joseph Muscat* and *University of Malta* may be constructed, with the relation label being Alma mater. This is the case with all data represented in the info-box.

Furthermore, in-depth relation analysis is achieved through the request of further data pertaining to the secondary entities. This is possible by recursively calling *getInfoBoxRels* passing secondary entities as parameters. In the above provided example, University of Malta will be explored by means of a further request to *Wikipedia* for its own info-box data. This one-level-depth recursive call for each entity allows us to extract interesting and meaningful relations not only between main and secondary entities, but amongst secondary entities themselves.

This recursive implementation results in a comprehensive

Figure 1: Wikipedia Info-Box for Joseph Muscat

understanding of each entity. The results of this function are returned in the form of a linked list relating primary entities to secondary entities with a relation label. This set of relations make up the majority of relations visualised within the graph. However, for the sake of thoroughness, further relations are extracted from the raw text data.

3.3 Processing of Raw Text

Article raw text saved within text files is of little use in its current form, and requires extensive processing before relations may be extracted from it. The *Stanford Core NLP Library* [13] was the library of choice for the handling and processing of text for the extraction of meaningful data. This library offers an extensive set of natural language tools, including tokenisation, part of speech tagging, co-reference resolution and named entity recognition. This library was imported into the program in the form of an executable JAR file, and was invoked by means of a process builder. The process builder executes the Stanford library through the use of the operating system command line, passing the raw text file as an argument to the NLP processor.

The results of this processing are stored in yet another text file so as to reduce the run-time of the algorithm on consecutive calls for the same main entity relations. The result file holds XML data describing key features of the raw text. Data from this complex file is then organised into a usable manner by means of the *extractInfo* function. This

function organises the data into manageable linked lists. A list of named entity tags is stored for each token, marking each token as an organisation, person, location or non-entity. This list will be used for the distinguishing of entities from non-entities, allowing us to construct meaningful relations between entities. A list of co-referring tokens is also constructed, linking different tokens that refer to the same entity. For example, in the following sentence:

"Joseph Muscat took office on Monday 11th March 2013. He and his wife Michelle Tanti..."

tokens He and his are listed as co-referring to the entity Joseph Muscat. This form of information is vital for the decoding of complex and ambiguous relations.

3.4 Relation Extraction

Multiple approaches are available for the extraction of relations from raw text. These range from unsupervised and supervised approaches, all the way the hand-written rules. Unsupervised approaches involve the use of generic relation extraction algorithms, which extract numerous, possibly meaningless relations from text. Supervised and hand-written approaches take a more customised approach to relation extraction, tailored specifically to a certain style of text. The latter of the two involve painstakingly formulating rules for the correct extraction of relations.

Out of the three above approaches, the unsupervised approach by means of the *Reverb* [14] algorithm was selected. This selection boiled down to the lack of gold standard training data for the training of a supervised algorithm for the relations expected to be found between the provided main entities. Most training sets and algorithms employed within the industry are more suitable for general or simple relation extraction, and where found to ignore salient potential relations. Furthermore, the manual coding of relation rules was found to be highly tedious, due to the lack of rigidity in which the *Wikipedia* articles of the different entities were worded and constructed. This moderate level of flexibility of *Wikipedia* articles is one adverse characteristic of this vast repository.

The *Reverb* algorithm was invoked in the same manner as the *Stanford Core NLP Library*, through the creation of a process builder, executing the algorithm through command line. The result is a string holding a list of relations extracted from the raw text, together with meta data such as accuracy ratings. This relational data is organised within the *extractRels* function, only retaining relations holding an accuracy rating above the user-specified threshold. By default, this threshold is set to 0.8.

3.5 Co-reference Resolution

Co-reference resolution is performed on the relations extracted through the unsupervised algorithm. Each segment of the relational triple is tested for co-reference, and is replaced with its head co-referring token. For example, the following triple:

<He, studied at, University of Malta>

is replaced with:

<Joseph Muscat, studied at, University of Malta>

This process standardises all relations, ensuring that each argument within the relation refers to the highest standard form.

3.6 Relation Trimming

The final step in the processing of raw text relations is the exclusion of insignificant relations. This procedure is completed by the *trimRelations* function, which loops through all relations for the current entity, removing any which fail one of the two tests.

First, each primary relation argument (left-hand side of triple) is tested to see whether it refers to the current main entity; as all relations should. If any given relation does not relate the main entity currently being considered, then it is removed. Secondly, each relation is tested to ensure that the secondary relation argument (right-hand side of triple) refers to a named entity. If this is not the case, then the relation is removed as it does not hold any useful relations. All relations which make it through these tests should hold salient information.

3.7 Graph Visualisation

Now that all the relations from the raw text and the infobox have been extracted, we must generate and visualise a relation graph. The Java *GraphStream* visualisation library [15] was implemented for the generation and depiction of the graph. This library provides a wide variety of tools and flexibility with style and design of the graphs, and therefore was considered ideal for this implementation. Nodes were created in the graph for the set of entities on either end of the relation triples. One node was defined for multiple mentions of the same entity, with co-reference resolution being applied to ensure that duplicate nodes referring to the same single entity are not created.

A directed edge between the nodes was then defined, with the direction depicting the primary relational argument as the source, and the secondary relational argument as the target node. This resulted in the linking of each separate main entity graph into one large graph linking all the main entities. Styles and graph display settings were tuned for the most beneficial visualisation, with main entity nodes being marked as red, and all nodes and edges carrying a label.

Further relation trimming was conducted here, with repeated relations (edges) between the same two entities being removed. A test was conducted on each node, ensuring that it is reachable from all of the main entities in the graph. A maximum hop count was also incorporated into this function, allowing the user to set the parameter for the maximum edge distance any node may be from all of the main entity nodes. This allows the user to generate graphs as general or specific as they like, providing added controllability.

3.8 Moralisation

Moralization is the process of converting an acyclic directed graph into a moral graph, which is an intermediate representation required for further observations, such as community detection. Moralization involves detecting all nodes which have a common child. Once a common child is found between two nodes, an unlabelled edge is added between the two parent nodes, and the edges connecting these parent nodes to their common child are converted to unlabelled edges.

This was performed by iterating through all nodes in the graph in a nested manner, testing whether any nodes have children in common. Newly added undirected edges were set to be marked in red for ease of observation and understanding. The invocation of function moralization results in the generation of a moral copy of the original relational graph.

3.9 Community Detection

Community detection is the process of finding and extracting maximal cliques from a relation graph so as to better understand the types and forms of relations present. Cliques are made up of entities (nodes) which are all mutually related, meaning that a direct relation between each entity in the community exists. This can be visualised in the form of a fully connected graph holding each member in the community. Communities are maximal cliques as they are the largest possible cliques - if any node outside of the current clique where to be added to the current clique, then it would no longer be fully connected and, thus, would no longer be a clique.

The community detection algorithm implemented in this system is the *Bron-Kerbosch* algorithm which was found to be more suitable than the *Girvan Newman* approach. 2 different functions were coded, with the second being favoured due to its improved efficiency. The first approach generated different combinations to check if any maximal cliques were evident before calling the community detection algorithm the extract communities. This approach made use of nested loops and also recursions with the community detection first starting with the full set of nodes in the graph. With every recursive jump, a node was deducted until a maximal clique was found. Although this showed successful results when tested on small graphs it took a considerable amount of time to execute on more complex network structures. This is why an alternative approach was sought.

The community detection algorithm implemented follows a recursive structure, recursively considering larger sets of nodes and testing whether they form a clique (community). Three sets, R, P and X are considered, with R holding the prospective community-forming nodes, P holding all neighbouring nodes which could possibly be considered in the current community, and X holding all nodes already considered in the current community.

The *communityDetection* algorithm is first called to initialise the variables and perform the first *Bron-Kerbosch* call. Within the *BK* function, if at any recursive level, P and X are found to both be empty, then the set of nodes in R is found to be a maximal clique and is saved as a community. If either P or X is not empty, then an iterative loop through all remaining nodes in P is called, recursively invoking *BK* with modified node sets P, R and X.

The result of this algorithm is a linked list of sets holding nodes forming different communities. These sets are passed on to the *community_visualisation* function which visualises each community by means of a new separate graph. These newly discovered communities allow us to better understand the business and relation of each of the main entities with other main and secondary entities.

4. RESULTS

The graphs on the following pages are some of the visualisations produced by the system upon execution. For this round of execution, the main entities considered were *Joseph Muscat* and *Simon Busuttil*, and the maximum step size from all main entities to secondary entities was set to 2. However, it may be noted that this system can operate with any number of main entities greater than 1.

Figure 2 depicts the graph visualisation for the directed relations between *Muscat* and *Busuttil*. As may be observed, the main entities are marked in red while other entities are marked as black nodes. This graph is a multi-graph, allowing multiple edges between the same two entities, with each edge having a different label. Interesting relations may already be understood, however, moralization and community detection is yet to be performed on the graph.

Figure 3 plots the moralized version of the relation graph, with newly added edges marked in red. As may be observed, all edges are now undirected, and we may already manually identify smaller communities within the graph. Certain segments of the graph are clearly denser than others, signifying the possibility of common relational entities between the main entities.

The final result of the system is depicted in the fourth figure, being just one of 6 communities detected. The following is a verbose listing of all the detected communities:

Community 1 of 3 nodes: National Order of Merit (Malta) - Joseph Muscat - Marie Louise Coleiro Preca

Community 2 of 10 nodes: Prime Minister of Malta - Labour Party (Malta) - Simon Busuttil - Louis Grech - Joseph Muscat - Marie Louise Coleiro Preca - George Abela - PietÃă, Malta - Malta - Lawrence Gonzi

Community 3 of 8 nodes: Simon Busuttil - Louis Grech - Joseph Muscat - Marie Louise Coleiro Preca - George Abela - University of Malta - Malta - Lawrence Gonzi

Community 4 of 3 nodes: Simon Busuttil - Louis Grech - Member of the European Parliament

Community 5 of 4 nodes: Labour Party (Malta) - Simon Busuttil - Malta - Parliament of Malta

Community 6 of 3 nodes: Joseph Muscat - List of Prime Ministers of Malta - Lawrence Gonzi

Figure 4 is the 3rd discovered community out of 6, holding a total of 8 nodes within the clique. A lot may be learnt from communities such as this, consisting of person, location and organizational entities. Considering the individuals forming part of this community, we may immediately understand that they are all politically related, meaning that they all took office or held a political role within close time frames. We may also conclude that all the mentioned politicians are related to Malta and attended University of Malta at some stage during their education.

Figure 5 is one of nine communities extracted from a relation graph considering all 8 political entities with a step size of 5. This community is interesting as it allows us to easily understand characteristics of some politicians through simple relations. We may observe that the above 4 politically related individuals all have a relation to the Catholic Church, signifying their belief in the Catholic faith. This is yet another example of information retrieval through relation extraction and community identification.

5. FUTURE WORK & CONCLUSIONS

The results achieved by the implementation outlined above have been considered satisfactory, with all major requirements working correctly, extracting and visualising meaning-

ful results. This being said, future work on this application could see deeper more meaningful relations being extracted by considering further data processing.

The full article text processing was found to yield little meaningful relations, mainly due to the generality of the *Reverb* algorithm and the problems encountered when filtering out meaningful relations. The training of a supervised relation extraction algorithm would allow for the finding of more suitable relations, enriching the end product of the algorithm.

The consideration of more than one source for data acquisition would also offer new relations which are possibly not present within other sources, such as *Wikipedia*. It would also be interesting to apply to algorithm to different, possibly biased, articles and data repositories, comparing the different visualisations produced from the different sources. For example, an observation in the portrayal of relations by one party news broadcaster, and the opposing party news broadcaster prior and during an election would most likely yield different, interesting communities.

Further improvements could also be applied to the visualisation process. As the graph progresses, more nodes and edges are evident in the network structure but in some cases the graph was harder to read. As paper [1] proposed, *Fisheye* could be used to help the user zoom/pan in and out as needed for an easier way to read and view the connections within the network structure. This method would have been employed to help solve such an issue.

The inclusion of a graphical user interface, allowing the user to manually specify and select a group of entities to consider, would improve the flexibility and user-friendliness of the system. This would allow added control of graph visualisation, making it easier for users to select and observe communities and relations which they are specifically interested in.

The use of a visualisation library allowed for the illustration of the final result in a simple, clear and easily understandable manner. Compared to the *Wikipedia* article, the graph visualization provided a summary of key information with the possibility to view and understand the most prominent content in a shorter amount of time.

This system has proven that the extraction, analysis and visualisation of possibly large sets of data allows us to understand underlying relations implied. This application has bettered our understanding of the principles of knowledge discovery, representation and management, covering a wide range of algorithms and approaches used within this field.

6. REFERENCES

[1] (1st June 2016). White paper: Cisco VNI Forecast and Methodology, 2015-2020. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.

[2] Google Search Statistics . Available: http://www.internetlivestats.com/google-search-statistics/.

[3] (13th January 2017). 36 Mind Blowing YouTube Facts, Figures and Statistics - 2017. Available: https://fortunelords.com/youtube-statistics/.

[4] Total number of Websites . Available: http://www.internetlivestats.com/total-number-of-websites/.

[5] Does Visualization Really Work? Here's Evidence That It Does. Available: http://expertenough.com/1898/visualization works.

[6] R. M. Tarawneh, P. Keller and A. Ebert, "A General Introduction To Graph Visualization Techniques," 2012.

[7] K. Ma and C. W. Muelder, "LargeScale Graph Visualization and Analytics," pp. 39, July 2013, 2013.

[8] J. Yang, J. McAuley and J. Leskovec, "Community Detection in Networks with Node Attributes"

[9] A. Clauset, M. E. J. Newman and C. Moore, "Finding community structure in very large networks"

[10] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis"

[11] A. Conte, "Review of the Bron-Kerbosch algorithm and variations"

[12] L. Bohlin, D. Edler, A. Lancichinetti, and M. Rosvall, "Community detection and visualization of networks with the map equation framework"

[13] Stanford Core NLP Java Library. Available: http://stanfordnlp.

[14] Reverb Java Library. Available: http://reverb.cs.washington.edu

[15] GraphStream Java Library. Available: http://graphstream-project.org/

7. APPENDIX

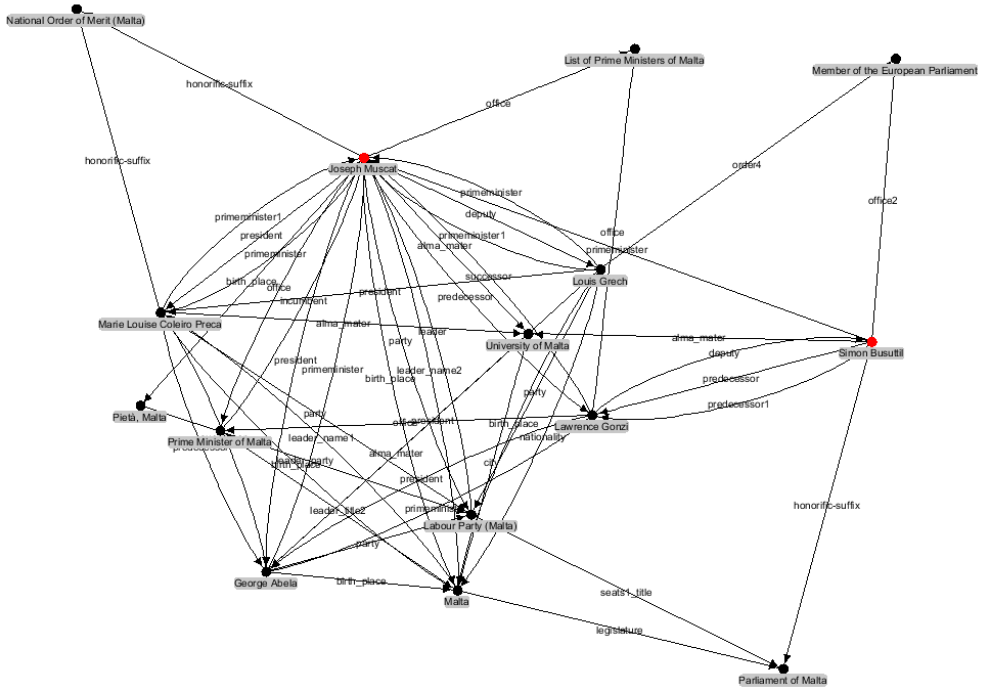

Figure 2: Relation graph for Joseph Muscat & Simon Busuttil

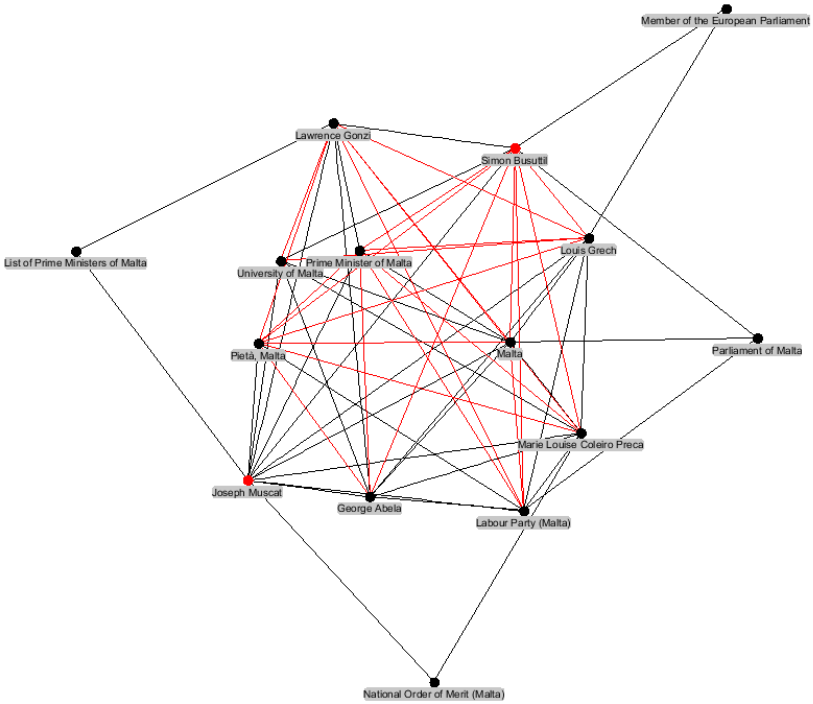

Figure 3: Moral graph for Joseph Muscat & Simon Busuttil

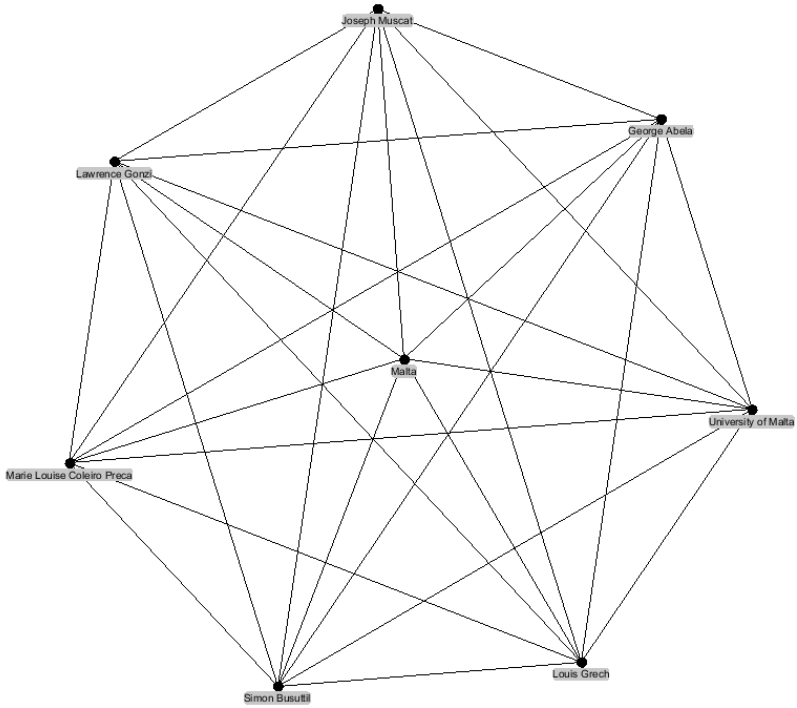

Figure 4: One of the discovered communities consisting of 8 entities

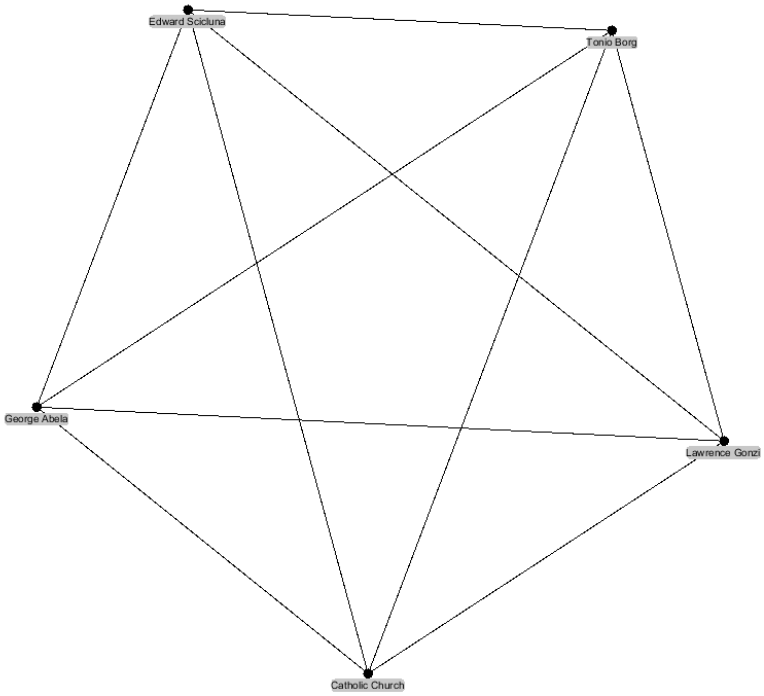

Figure 5: Another one of the discovered communities consisting of 5 entities