

Final Year Project Report

VSMing Newgrange

Ryan Lacerna

A thesis submitted in part fulfilment of the degree of

BSc (hons) in Computer Science

Supervisor: Prof Mark Keane



UCD School of Computer Science and Informatics
College of Science
University College Dublin

March 12, 2014

Table of Contents

Abstract	2
1 Introduction	5
1.1 Aims	5
2 Background Research	6
2.1 Symbols	6
2.2 The Vector Space Model (VSM)	8
2.3 Motivation for Vector Space Models for symbol frequency analysis	10
2.4 K-Means Clustering	11
3 Design	13
3.1 Program structure	13
3.2 Design specifications	14
3.3 Visualization	14
3.4 Work Breakdown Structure	15
4 Implementation	17
4.1 data.csv	17
4.2 Frequency model class	18
4.3 Vector space model class	18
4.4 Additional features	19
4.5 Problems	20
5 Analysis	21
5.1 Results	21
6 Summary	24
6.1 Critical analysis	24
6.2 Conclusion	24
6.3 Future work	25

Abstract

In the past, there have been several analysis that were carried out on Neolithic (Stone age) tombs based on percentages of coverage on certain symbols on given stones, but an extensive and complete analysis has yet to be carried out. Vector space model (VSM for short) is one of the most effective models in the information retrieval systems; this metric is a statistical measure with the purpose of evaluating how important a word is to a collection of documents or corpus. In this project the aim is to adapt the vector space model to measure the similarity between Neolithic sites and tombs around the Irish Sea by analysing the frequencies of their symbols. This work also highlights alternative applications of similarity computations in how it they be used in comparing archaeological artefacts.

Keywords: Vector Space Modelling (VSM), Similarity, Frequencies, Symbols.

Acknowledgments

The author of this thesis would like to address his most profound thanks to the project supervisor Prof. Mark Keane of the School of Computer Science and Informatics, UCD for investing his time in giving invaluable guidance, feedback and support throughout the process of this project.

This work owes much to Dr. Guillaume Robin of Department of Archaeology and Anthropology, University of Cambridge. The database of symbols he provided was an indispensable resource to this project.

This project have benefited from the advice and opinions of colleagues in School of Computer Science and Informatics, UCD. I give my best regards to them and their plan for the future.

Project Specifications

Subject: Vector Space Modelling (VSM) Analysis of Symbols

Project Type: Design and Implementation

Software Requirements: Ruby programming language

Hardware Used: PC

Objectives:

- Research previous analysis of symbols
- Develop a program that groups these symbols according to their boundaries and classifying unique symbols.
- Develop a database of these symbols
- Develop the program that will analyse the frequencies of these symbols using VSM
- Elaborate other ways in which similarity computation could be used to compare archaeological artefacts.

Aim: The main aim of this project is to develop a Vector Space Model (VSM) analysis of symbols with a view of discovering similarities between different tombs and sites.

Chapter 1: Introduction

The Neolithic period, around 3400 BC (traditionally considered as the last part of the Stone Age), was a period of development of human technology [14]. They created passage tombs (or passage graves) which consist of narrow passages made of large erected stones called orthostats which then lead to a small chamber(s) at the end. The passage tombs are then enclosed by an array of large slabs or kerbstones around the base of these tombs. These large stones are highly decorated with megalithic carved art, but the meanings of these symbols are still a large mystery to archaeologists, but some speculate that it must have some religious significance [9].

One of the best examples of passage tombs and megalithic art are located in a site in country Meath which consists of 3 large passage tombs Newgrange, Dowth, Knowth and other smaller satellite tombs. These sites consist around 60 percent of all passage tomb art around Western Europe [15].

These sites are among others in the British Isles and Western Europe which consist of various symbols such as; Lozenges, Chevrons, Concentric circle, Serpentine and one of the most unique which is can only be found at Newgrange; the Tri-Spiral. Although these symbols are grouped in categories and the count of each symbols for each tombs are recorded, the passage tomb art around the Irish Sea is considered to be abstract and unstructured [10].

Several current analysis have been carried out based on percentages of coverage of certain symbols on given stones, but an extensive and complete analysis is yet to be carried out.

1.1 Aims

Expanding from previous analysis of symbols, the aim of this project is to develop a Vector Space Analysis (VSM) program to analyse frequencies of symbols around the Irish Sea, with the comparative aim of finding similarities between sites and tombs together with the goal of discovering the use of VSM and similarity computation in comparing archaeological artefacts.

The project commenced with an extensive background research on the previous analysis of symbols that were conducted by experts in the field, and then a thorough investigation of the VSM and how it works. This paper will outline the background research that were conducted, a detailed design of the program, the detailed outline of implementation of each classes and methods, then an illustration of empirical results of the program and finally the conclusions and future work.

Chapter 2: Background Research

This chapter is a review of the background material. This includes a history of the symbols and an inspection of previous analysis, then moving on to the research of the vector space model discussing its history, technique and applications for calculating similarity such as to be seen in this project, and finally a look at the K-means algorithm as an extension of our program to investigate other ways in which to use similarity computation.

2.1 Symbols

2.1.1 History

The passage tombs around the Irish Sea form part of a group of monuments constructed during 4th millennium in the Western Part of Europe [4]. The purpose of this funerary architectures is to contain the remains of reserved number of people, who has an obvious dominating role in their social group. From Iberia to the British Isles, this type of monuments all inherit a circular tumulus(see Figure 2.1) structure consisting a central chamber with an access passage whose walls are made with large slabs of stones. The large slabs of stones are then used as canvas for carved art (see Figure 2.1)



Figure 2.1: Neolithic passage tomb(Left) and Neolithic carved art (Right)

Around the Irish Sea, these carved arts are possessed mainly of geometric features, which differentiate it from Western France and Iberian symbols which compose mainly of naturalist representations such as objects, human beings, and animals.

2.1.2 Previous analysis of symbols

There have been a few thesis that were conducted on research of parietal (cave wall) art. E. Shee Twohig have dedicated her academic works to the documentation and the analysis of the carving placed outside the Boyne Valley[5] whereas M. O'Sullivan was concentrated in the parietal art of Knowth necropolis[6].

This project will use Robin, G.'s thesis *Neolithic passage tomb art around the Irish Sea Iconography and spatial organisation, Volume 1*.

"To seek the law of the signs, it is to discover things which are similar"

Robin, G. states in his thesis, that this quote above by Foucault, M. [11] from its initial context, summarizes well the original reasoning of his research. This project is based on frequencies and similarities of symbols, so it is important to have an understanding how symbols were previously analysed in order for this project to deliver a substantial and meaningful result.

For this project certain details need to be extracted from previous analysis data; different types of symbols, which tombs and sites they are located, and most importantly the frequencies of these symbols.

Following the thesis symbols were grouped according to their boundaries which can be seen below:

Circular and Semi-circular signs:

- dots and cupmarks
- circular signs
- spirals
- arcs
- radiate circular signs
- radiate semi-circular signs

Angular signs:

- chevrons
- triangular signs
- quadrangular signs
- scalariform signs

Combinations of symbols were also grouped::

- combination of circular signs
- combination of arcs
- combination of chevrons
- combination of triangles
- combination of lozenges
- combination of meandering signs

The locations of symbols are also specified:

- Dh.K13 which translates: Dowth(Dh), Kerbstone 13(K13)
- Ng.K17 which translates: Newgrange(Ng), Kerbstone 17(K17)
- KhE.Or11 which translates: Knowth E(KhE), Orthostat 11(Or11)

There are many more translations of these symbol locations highlighted in the thesis.

Most importantly the frequency of symbols, here are examples from the thesis:

- Dots and cupmarks are set to appear on 28% of the carved stones
- Circular signs appeared in 282 stones, which is 45% of carved stones
- Chevrons signs appeared in 170 stones, which is 26.8% of carved stones

Although the thesis is detailed, some frequencies were missing, and a complete vector of frequencies is needed in order for the VSM program to give a substantial result. An alternative approach is required, and Robin, G. provided a database of all the 634 stones which the author used for this thesis. This contained the details of the location of each symbols, their type and occurrence. This database is precisely what the program needed as input as it provided a full set of vectors to use, details on this will be discussed later in this paper.

2.2 The Vector Space Model (VSM)

Vector space model or term vector model is one of the most effective model in the information retrieval system, it is an mathematical model for representing text documents and any objects in general as vectors of identifiers, such as for example index terms, in other words, the model creates a space in which both documents and queries are represented by vectors. This is used in information retrieval, information filtering, indexing and relevance rankings and its first use was in the SMART Information Retrieval System [1]. SMART pioneered many of the concepts which are used in modern search engines. The success of the vector space model for information retrieval has inspired many researchers to expand the VSM to various semantic tasks in processing natural language, with impressive results. Next is an exploration of the history of VSM to find out more on how this popular model came to be.

2.2.1 History

Gerard A. Salton was a Professor of Computer Science at Cornell University and is regarded as the leading computer scientist engaged in the field of information retrieval during his time. While he was at Cornell his group developed the SMART Retrieval System, which he initiated at Harvard. SMART (System for the Mechanical Analysis and Retrieval of Text) is more than just an IR system, SMART was the working result of Salton's theories and the experimental environment in which these theories were tested and evaluated and many influential concepts in IR were developed as part of research of the SMART system, including the Rocchio Classification relevance feedback and the vector space model.

In 1974 and 1975 Salton published several influential papers on a theory of indexing and methods for selecting terms from documents and attaching numeric weights to them. This was called the *Term discrimination value* model (TDV for short), and would prove to be critical not just because an automated indexing principle was formulated in this model but also of its impact on the information retrieval community's perception of what to be known as the vector space model.

The next symbolic evolutionary stage of the vector space model came to be perceived has become evident in 1979. The year Salton published an article the *Journal of Documentation* (JDoc for short) titled *Mathematics and Information Retrieval*. This article was the first since the 1968 text *Automated Information Organisation and Retrieval* that Salton published to discuss the issues of modelling in depth, it is important for two reasons:

- Its first time Salton referred to the vector space model as an Information retrieval model in print.
- Salton described orthogonality assumption in this article for the first time.

Obtaining knowledge of the VSM introduction and history this paper will now progress to the VSM algorithm, demonstrating its technique.

2.2.2 The VSM Algorithm

The VSM creates a space in which both documents and queries are characterized as vectors. For a fixed collection of documents, a m collection dimensional vector is generated for each document and each query from the sets of terms with weight associated, where m is the number of unique terms in the document collection. Then, a vector similarity function, just like the inner product, can be used to compute the similarity between query and a document [2].

In Vector Space Modelling, the weights that are associated with the term are calculated based on TF IDF:

- Term Frequency(TF), f_{ij} , which is the number of occurrences of term v_j in the document x_i ; then,
- Inverse Document Frequency (IDF), $g_j = \log(N/d_j)$, where N is the total number of documents in the collection and d_j , is the number of documents containing term v_j .

where m is the number of unique terms in the collection of document. The query v_j and the document weight w_{ij} are:

$$w_{ij} = f_{ij} \quad w_{ij} = f_{ij} \cdot \log(N/d_j)$$

and

$$v_j = \begin{cases} \log(N/d_j) \\ 0 \end{cases}$$

v_j is a term in q otherwise

$$sim_{vs}(q, x_i) = \frac{\sum_{j=1}^m v_j \cdot w_{ij}}{\sqrt{\sum_{j=1}^m (v_j)^2 \cdot \sum_{j=1}^m (w_{ij})^2}}$$

The similarity $sim_{vs}(q, x_i)$, between a query q and a document x_i , can be defined as the inner product of the query vector q and the document vector x_i

Once the TF-IDF values were achieved, the popular approach is to use cosine similarity to compare the query with the document, but for this project the aim is to compare each document with one another with the aim of finding out which documents are similar. In the project's case, pair wise similarity will be used.

2.2.3 Pair wise similarity

Pair-wise similarity pairs each document with every other document giving each pair a numeric score indicating the similarity between paired documents. The higher the score, the more similar the paired documents are.

The similarity between document $D1$ and document $D2$ can be defined as the inner product of the document $D1$ and the document $D2$

$$D1 = w_{11} * w_{12} * w_{13} * w_{14} * w_{15} \dots * w_{1n}$$

$$D2 = w_{21} * w_{22} * w_{23} * w_{24} * w_{25} \dots * w_{2n}$$

$$sim_{vs}(d_1, d_2) = \sum_{i=1}^m w_{1i} * w_{2i}$$

2.3 Motivation for Vector Space Models for symbol frequency analysis

VSM have several attractive properties. VSMs automatically extract knowledge from a given corpus, so they require less labour than other approaches, such as hand coded knowledge base and ontologies. It performs well on tasks which involved measuring the similarity of meaning between documents, phrases and words. Most search engine use VSM to measure similarity between document and query[12], and the leading algorithms for measuring semantic relatedness also use VSM[17][18][19].

VSMs are especially interesting due to its relation with the distributional hypothesis and related hypothesis; *Words that occur in similar contexts tend to have similar meanings* [20][21] If the words have a similar row vectors in the word context matrix, then they tend to have sim-

ilar meanings. Efforts to apply this abstract hypothesis for concrete algorithms for measuring the similarity of meaning usually lead to matrices, vectors and higher order tensors. This intimate connection between the VSM and distributional hypothesis is a powerful motivation for taking a close look at VSMs.

Not every use of vectors and matrices count as VSMs. For the purpose of this project, The defining property of VSMs will be taken and that the values of elements in the VSM must be taken from event frequencies, such as the number of times that the given symbol appears in a given Site/Tomb.

2.4 K-Means Clustering

K-Means is arguably the most popular method for data analysis and one of the simplest unsupervised learning algorithms that solves the popular clustering problem. Clustering means classifying objects into different groups, or in other words, partitioning data sets into subsets(clusters), so that each of the data in the subsets share common traits, more often according to their defined measure of distance [3].

k-Means is an algorithm that cluster n objects based on attributes into k partitions, where $k < n$, it is similar to the expectation-maximization algorithm for mixtures of Gaussians function in the way they both attempt to search for the centre of clusters in the data [3]. It assumes that object attributes forms a vector space.

In this project, an expansion of the VSM is integrated to the k-means algorithm to find out which Tomb/Sites belong in the same group based on the quantified result of their symbols given by our VSM.

2.4.1 History

Cluster analysis emerged as a dominant topic in the 1960s up to the 1970s when the monograph *Principles of numeric taxonomy*[22] motivated the world-wide research on clustering methods and thereby initiated publications of a broad range of texts such as *Les bases de la classification automatique*[23], *Mathematical taxonomy* [24], *Cluster analysis for applications*[25] and many more, with the consequence that the primitive methods and problems of clustering became well-known in the broad scientific community, in data analysis, statistics, and in applications.

The term *K-Means* was first used by J MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1957. The standard algorithm was first proposed by Stuart Lloyd in 1957 as an approach for pulse code modulation, even though it was not published outside of Bell labs until 1982. In 1965, E W Forgy published essentially the exact method which is why it is often called Lloyd-Forgy. A much efficient model was introduced and published in Fortran by Wong and Hartigan in 1975, 1979.

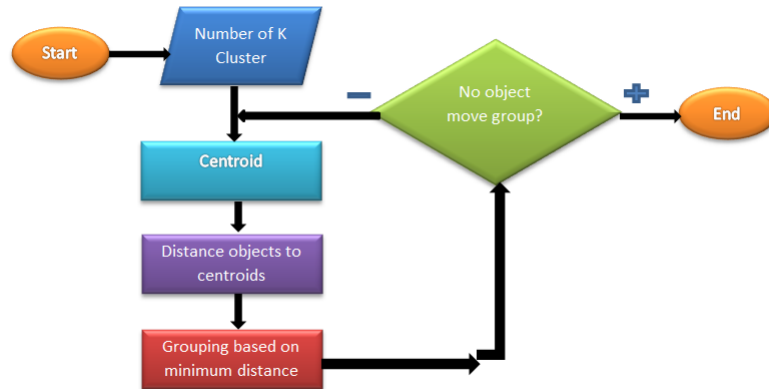
2.4.2 K-Means Algorithm

In this section is a brief description of the K-Means algorithm. The number of clusters k is assumed to be fixed in K-means. Let k prototypes (w_1, \dots, w_k) be initialized to one of the input n patterns (i_1, \dots, i_n) . Therefore,

$$w_j = i_l, j \in \{1, \dots, k\}, l \in \{1, \dots, n\}$$

C_j is the j cluster whose value is a high disjoint input pattern subset. The quality of clustering is determined by the error function:

$$E = \sum_{j=1}^k \sum_{i_l \in C_j} |i_l - w_j|^2$$



Step one: Start with a decision of the value $K =$ number of clusters.

Step two: Add any initial partition which classifies the data to K clusters, assign the training samples randomly or systematically:

1. Take the first K training sample as a single element clusters.
2. Assign each of the remaining $(N - k)$ training sample to the cluster with the nearest centroid. At the end of each assignment re-compute the centroid of the gaining cluster.

Step three: Take each of the sample in sequence then compute the distance from the centroid of each cluster. If the sample is currently not in the cluster with the nearest centroid, switch that sample with that cluster, then update the centroid of the cluster which gains the new sample and also the cluster that loses that sample.

Step four: Step 3 is repeated until convergence is achieved, which is until a pass through the training sample causes no new assignments.

Chapter 3: Design

This chapter formally introduces the project, providing a detailed specification of the program structure, design used and also discussion of the re-design and implementation issues that had been considered. It outlines the use of certain technologies and their appropriateness of the task. The chapter also discusses the design issues from a software perspective; such as software methodologies that were adhered to as much as possible; methodologies such as Object Oriented Design. Finally the chapter closes with comments pointing to the design and its effects.

3.1 Program structure

As stated previously in the project specification, the aim was to design and implement a vector space model analysis of symbols. More formally, the objective was for a software programme solution that will take the frequencies of tombs and sites and calculate their similarities using the vector space model algorithm. In order to achieve this goal, the program design operates in sequence at run-time, and represents the flow of data through the system. Figure 3 Shows a UML diagram showing a breakdown of the system and its components.

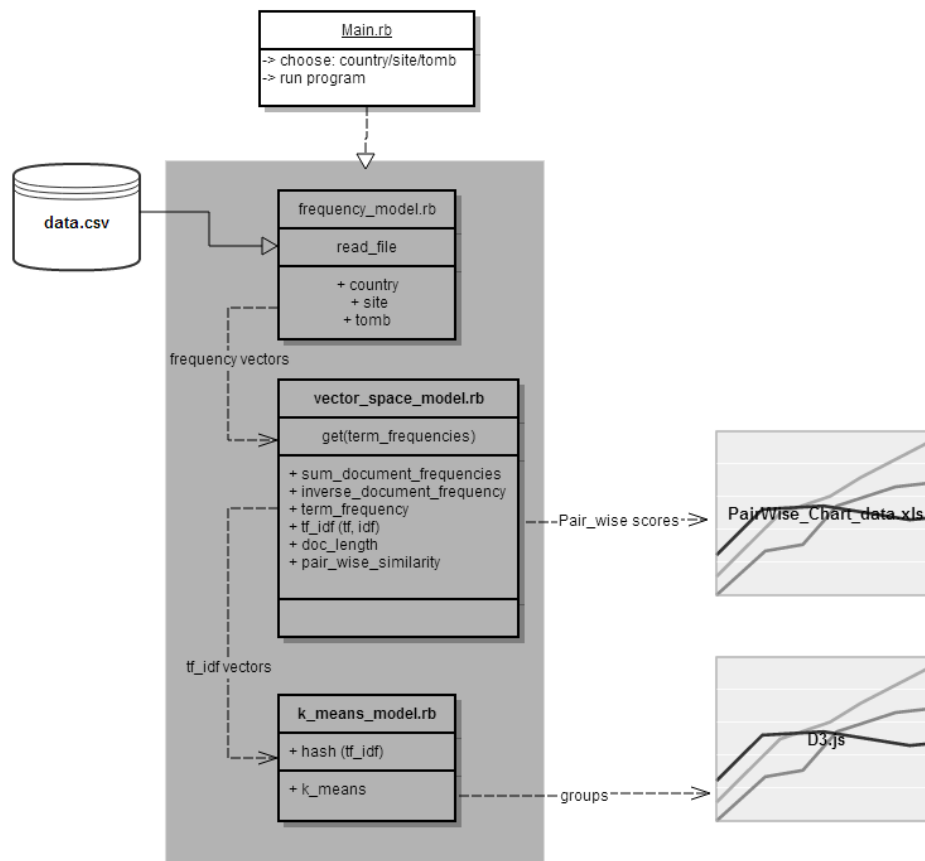


Figure 3.1: A high level illustration of the breakdown of system component delegation

3.2 Design specifications

The project is written entirely in the Ruby programming language using the RubyMine IDE. Ruby is a reflective, dynamic general purpose programming language [13]. In Ruby, variable declarations are not needed, variables are not typed, the syntax is very consistent and simple, and it also has automatic memory management [16]. This is a type of interpreted scripting language that allows quick and easy programming. This allows the programmer to focus more on the task and result rather than the programming aspect, all this combined makes Ruby suitable for this project.

All the project's applications are coded following the object oriented design concept. All components in the system are derived as objects and can be accessed by other components in the system. It also made sense later in the implementation, as the classes in the system each have a predefined objective and workload. This also made the design of separate processes more feasible and a natural goal to work on. The design provided a simple means to implement the system; there were a few re-implementations from the original specification and also a few software methodology violations with it.

- Although the program was originally intended to use the symbol frequencies quoted from the thesis of previous analysis of symbols, frequencies did not appear in some locations which threatened the substantial result of the program, an excel database was given by the author of the thesis Robin G.[7], which contained complete symbols occurrences, location and type. This excel database was then translated into Comma-separated values (CSV) and used as input.
- The original project specification was to create a database using *SQLite3* which will then be used by the VSM. Using the `frequency model` class and the new input re-implementation mentioned above, a database was no longer required as the `frequency model` class can take input the CSV file and produces an array of vectors which the VSM can use. This new design resulted in better efficiency and less code.
- Some methods violated the Don't Repeat Yourself (DRY) principle; such methods are in the class `frequency model`. This implementation was designed purposely for the user to freely choose from the Main class to which group (country, site or tomb) of frequencies he/she would like to work with, this will be discussed in more detail in the implementation chapter.
- The original plan of hard-coding K-means was unrealistic time-wise, so a Ruby gem was used for its implementation, this strategy saved valuable time thus still providing empirical results.

Taking all this into account, there were other minor re-design implementations at the early stages of the project which were mostly factorisation, the reason was for the program to be more object oriented and to work with less code making it easier to manage and correct calculation mistakes.

3.3 Visualization

A few visualization tools were tested before coming to a conclusion to which one to use for visualising the data. The first tool that was used was *Highcharts*. *HighCharts* is library

written purely in *JavaScript* and *HTML5*, offering intuitive interactive charts. This library supports many different types of graph such as; line, areas, column, bar, pie and many more. For this project the bar graph was used in the early stage but was proven to be an inadequate way for visualising similarity data. Another tool was needed that can provide high customizability to suit this project’s needs.

3.3.1 Microsoft Excel

For visualizing similarity data generated, a type of scatter plot graph, which can represent distance of similarity between two entities. For this project Excel was chosen, it is powerful spreadsheet application developed by Microsoft. It offered several choices for graphs and provided high customisability. This will visualize the data for the pair-wise similarity.

3.3.2 D3.js:A JavaScript Visualization tool

In order to visualize the data for K-Means clustering, a different type of tool was needed, A tool that can visualize the grouping that our clustering algorithm has generated. *D3.js* is a *JavaScript* library used for manipulating documents based on the data. It allows the user to bind data to a Document Model Object (DOM for short), and then applies data driven transformations to the document [6]. This visualisation tool created a beautiful circle packing chart which illustrated the document grouping perfectly.

3.4 Work Breakdown Structure

The Project design specified 5 months of research and implementation. The first months of the project after initiation were dedicated to intensive background research. A lot of time was spent investigating other solutions, to acquire better knowledge of the problem and how the task has been attempted before. Figure 3.3 Illustrates a Gantt chart showing clearly the milestones and tasks. This breakdown of work allowed contingencies in the project, as it allowed time management and avoiding pitfalls such as deadlines and over-running of readings. This Work Breakdown Structure (WBS) was an important consideration in the design of the project, as it controlled the flow of the project and also define a strict plan, and allowing for a steady and incremental approach taking the project from its specifications through implementation.

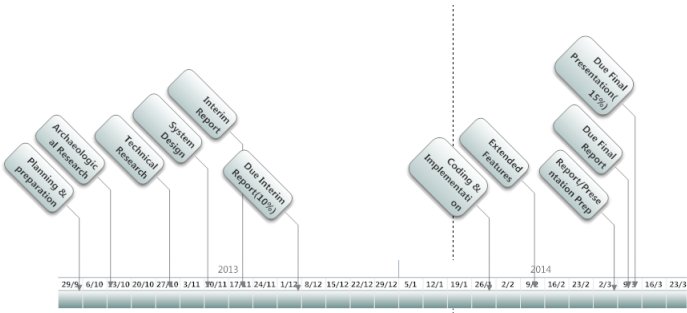


Figure 3.2: Timeline view of project milestones

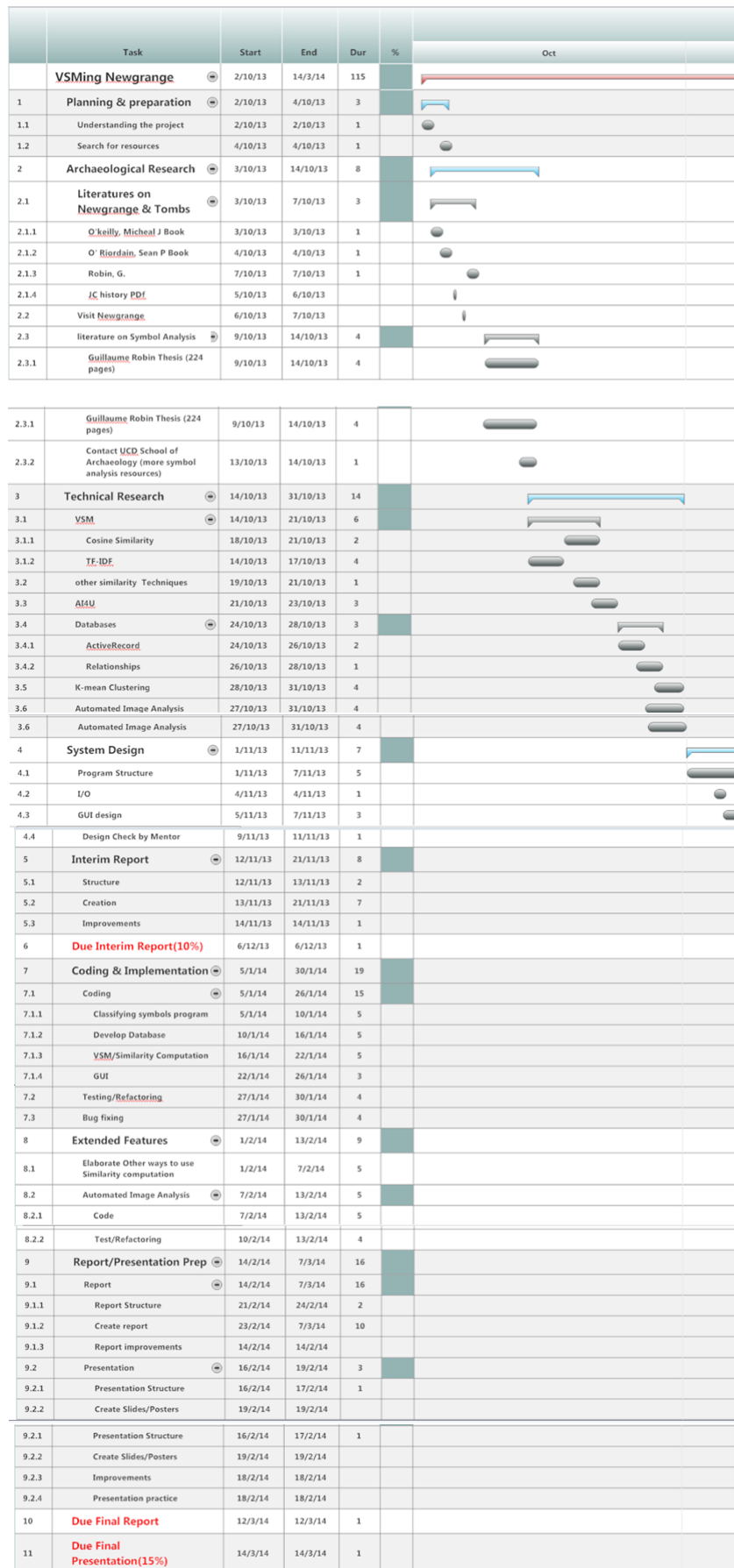


Figure 3.3: Gantt chart illustrating the work breakdown structure and milestones of the project

Chapter 4: Implementation

Continuing from the previous chapter on the project's design, the goal of this chapter is to detail how the application was implemented once the basic structure of the projects was designed. This section will describe how the classes and methods were developed for the application. It will also highlight any problems that were encountered during the development and how they were resolved.

4.1 data.csv

Before development began it is necessary to choose a substantial input to use for the program calculation, as mentioned in the previous chapter that using the frequencies quoted straight from the thesis was impractical as it lacked data to complete a full vector. An Excel spreadsheet(see Appendix A.1) that contained details of symbol's occurrence, location and type was then used and translated to CSV. Figure 4.1 illustrates a screen shot of a part of the CSV file being used after its been converted and customized for the use of this project.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Meath	Ardmulchan	Ireland	1	1		1					1		1	1
2	Meath	Ardmulchan	Ireland	1										1	1
3	Meath	Ballinvally 4	Ireland	1	1		1								
4	Meath	Baltinglass	Ireland		1		1								
5	Meath	Baltinglass	Ireland		1		1								
6	Meath	Baltinglass	Ireland			1									
7	Meath	Baltinglass I	Ireland											1	
8	Meath	Baltinglass II	Ireland		1	1									
9	Meath	Baltinglass II	Ireland		1	1									
10	Cavan	Banagher	Ireland		1							?		1	1
11	Antrim	Carnanmore	Ireland		1		1								1
12	Antrim	Carnanmore	Ireland	1											
13	Kildare	Castledermot	Ireland									1			
14	Cork	Clear Island	Ireland			1				1					1
15	Meath	Clonasillagh	Ireland	1	1		1					?		1	
16	Sligo	Cloverhill	Ireland	1	1		1								
17	Sligo	Cloverhill	Ireland			1					1				
18	Sligo	Cloverhill	Ireland		1	1					1				
19	Sligo	Cloverhill	Ireland	1	1	1	1				1			1	
20	Meath	Cregg	Ireland							1					
21	Meath	Dowth	Ireland		1	1	1								
22	Meath	Dowth	Ireland	1											

Figure 4.1: screen shot of data.csv

Looking at Figure 4.1. the CSV file consists of a table of 15 columns, and a total of 634 carved stones were integrated into this inventory. The first three columns of the table indicate the site, tomb and the name of the country, the next twelve columns indicate each symbol's presence '1' or '?' indicate presence and blank indicate absence of the symbol of reference. Using the occurrences of each symbol's presence the program can calculate the frequencies.

4.2 Frequency model class

It was mentioned earlier in the project specification that the VSM needs to analyse the frequencies of symbols. Before the VSM could work, frequencies first need to be calculated from the presence or absence of symbols from a given site or tomb. The `frequency model` class reads from the CSV data file then parses through row by row searching for symbol presence(see Appendix A.2), once symbol presence is found in the document it adds it into a vector taking in to account the name of the location (e.g. Ireland) where the symbol was found. This process is repeated until the entire data file has been exhausted and all frequencies are recorded from the CSV file. Figure 4.2 shows a sample output from a vector showing the total frequencies found for the particular document (e.g. Ireland).

```
["Ireland", 169, 297, 147, 332, 49, 22, 172, 68, 135, 74, 203, 215]
```

Figure 4.2: sample output of total frequencies found for Ireland

4.3 Vector space model class

Following from the specification, the main aim of this project is to create a VSM analysis. Having obtained the Term frequency (TF) vectors from the `term frequency` class, the algorithm formulated in section 2.2.2 can be initiated.

- Having obtained TF, the next part in the program is to calculate the Inverse Document Frequency (IDF) to do this, the program is initiated by the `sum document frequency` method(see Appendix B.1), this calculates the frequency of documents that contains an occurrence of a symbol, this counts how many documents contains a symbol of reference. Once the document frequency is acquired, the `inverse document frequency` method(see Appendix B.2) then uses the *IDF* formula (see section 2.2.2) to give us the IDF values.
- To normalize the *TF* values, the `term frequency` method(see Appendix B.3) acquires the TF values gathered earlier and normalizes the data using $\log(TF) + 1.0$ to give us our normalized *TF* values.
- The `tf idf` method(see Appendix B.4) then acquires the *TF* and the *IDF* and performs a matrix multiplication of the two vectors with respect to the *TF-IDF* formula. (see section 2.2.2)
- Now that TF-IDF is achieved, the program can initiate calculating similarity metric of each document (e.g sites). Earlier in the development the wrong formula was used to calculate the similarity; Cosine similarity was used, but this algorithm's purpose is to compare a query with documents. In this project no queries are used but instead compare all documents with each other. To do this, Pair-wise similarity algorithm (see section 2.2.3) is adapted, this method(see Appendix B.5) take the first vector and compares it with the second vector by using the similarity metric formula which gives the paired document a score, this process is repeated until the first vector is scored with every other vectors in the matrix, the second vector then uses the same approach and so on until every vector in the matrix has been paired and given a metric score. Figure 4.3 shows a sample output that illustrates the pairing scores.

```

"Pair wise similarity::"
["Ireland", "Scotland", 6.49]
["Ireland", "Wales", 3.98]
["Ireland", "England", 3.44]
["Scotland", "Wales", 2.0]
["Scotland", "England", 1.76]
["Wales", "England", 1.06]

```

Figure 4.3: Sample outputs of pair wise scores

4.4 Additional features

To elaborate other ways in which to use similarity computation for analysing symbols, an extra feature is implemented to the VSM program. In this section is a discussion of how this feature is implemented and the technique in which it was visualized.

4.4.1 k means class

In order to elaborate another way to use similarity computation for analysing symbols, the K-Means clustering algorithm(see section 2.3) is used. As mentioned earlier in the design chapter hard coding the k-means algorithm would be unrealistic time-wise, so for this project the *kmeans*(0.1.1) gem was used from *RubyGems.org*. Ruby gem is a package manager for Ruby that provides standard format for distributing ruby libraries.

This gem takes in a Hash table of Hash tables as input for its algorithm. In order for the program to receive the array of TF-IDF value vectors generated by the VSM, a hash of hashes was created for the TF-IDF values(see Appendix C.1). It is then fed into the K-means program(see Appendix C.2).

Able to acquire the TF-IDF values using hash tables created, the number of centroids then needs to be specified and with it the number of loops. The code sample below marks where the number of centroid and loops needs to be implemented.

```

@result = Kmeans::Cluster.new(vector, {
  :centroids => 4,
  :loop_max => 100
})

```

Note: the grouping calculations in K-means gives different results for each run, multiple runs needs to be applied in order to achieve accuracy of results.

4.4.2 flare json and index html

In order to visualize the groups accumulated by the K-means program a certain type of visualization tool is to be used to ensure good visualization of the data. The *flare.json* and *index.html* is an open source JavaScript visualization tool taken from the *d3.js* website.

The data gathered from K-Means is manually scripted in to the *flare.json* file, once the data have been integrated, the data can then be visualized by running the *index.html*, thus

giving us the visualized data of K-Means.

4.5 Problems

At the late stage of the implementation few problems arose, Firstly a part of the algorithm that was implemented to measure similarity was not the one required for the specification of this project, this algorithm was called Cosine Similarity. The algorithm needed was one that will compare each document with every other document. With further research, the right metric (see section 2.2.3) was found and implemented, thus giving the appropriate results.

During implementation of the `k means model` class, the algorithm had shown to generate positive results when dealing with small vectors such as *country*. Although once a larger vector was used the program struggled to conclude a solid grouping. To work around this issue the program was run 60 times while recording each concluded grouped value, each results are then compared to one another and the most frequent occurrence is then used. Another issue with the *kmeans* gem was that the numeric values within the vector that needs to be input to the program needed to be above 1.0, anything less than the value is assumed by the program to be in the same group, a work around this issue is to create new method which updated all values(see Appendix B.6), thus making every numeric values in the input vector above 1.0, thus improving calculation to find groups.

Chapter 5: Analysis

Building up from the previous chapter on implementation, this chapter focuses on the presentation of results.

Note: Some results will be illustrated using the country vector for easier viewing of results, this vector is smaller than the other two vectors(site, tombs) thus making it easy to interpret. On the other hand the site vector is used to view the Pair-Wise similarity results, while tomb will be used for the K-Means.

5.1 Results

Each step in the calculation is presented to keep track of any changes made during any alternations in the program formula. Figure 5.1 shows the normalized term frequency (TF) values of symbols for each country. Figure 5.2 shows inverse document frequency (IDF) values, in this output it shows that the rarer the symbol occurs in the collection the higher its weight, while the more frequent the symbol appears in the collection the less weight is given.

```
"TF:"  
["Ireland", 5.136, 5.697, 4.997, 5.808, 3.912, 3.135, 5.153, 4.234, 4.913, 4.317, 5.318, 5.375]  
["Scotland", 1.792, 1.099, 1.792, 1.609, 0.0, 0.0, 2.773, 1.099, 2.303, 0.0, 1.946, 0.0]  
["Wales", 0.693, 0.0, 1.946, 0.693, 0.0, 0.0, 1.609, 0.0, 1.609, 0.693, 0.0, 1.386]  
["England", 1.792, 1.946, 1.946, 1.792, 0.0, 0.0, 1.386, 0.0, 1.386, 0.693, 1.099, 0.0]
```

Figure 5.1: Program output of normalized Term Frequency (TF) in countries

```
"IDF      [0.0, 0.0, 0.125, 0.0, 0.0, 0.602, 0.602, 0.0, 0.301, 0.0, 0.125, 0.125, 0.301]"
```

Figure 5.2: Program output of the Inverse Document Frequency (IDF) in countries

Figure 5.3 shows the output values after multiplying the TF with IDF. The TF-IDF value increases proportionally to the number of times the symbol appears in the document (country, site or tomb), but is then offset by the frequency of the symbol in the entire collection.

```
"TF*IDF:"  
["Ireland", 0.0, 0.0, 0.62, 0.0, 0.0, 1.89, 3.1, 0.0, 1.48, 0.0, 0.66, 0.67]  
["Scotland", 0.0, 0.0, 0.22, 0.0, 0.0, 0.0, 1.67, 0.0, 0.69, 0.0, 0.24, 0.0]  
["Wales", 0.0, 0.0, 0.24, 0.0, 0.0, 0.0, 0.97, 0.0, 0.48, 0.0, 0.0, 0.17]  
["England", 0.0, 0.0, 0.24, 0.0, 0.0, 0.0, 0.83, 0.0, 0.42, 0.0, 0.14, 0.0]
```

Figure 5.3: Program output of the TF-IDF values in countries

Lastly figure 5.4 illustrates the pair wise similarity results, depicting the similarity score when a given document (country) is paired with another document. The higher the score, the more similar the pair. Looking at the output below, it is easy to depict that Ireland-Scotland have the highest paired metric score, thus making it the two most similar documents in that matrix.

```

"Pair wise similarity:":
["Ireland", "Scotland", 6.49]
["Ireland", "Wales", 3.98]
["Ireland", "England", 3.44]
["Scotland", "Wales", 2.0]
["Scotland", "England", 1.76]
["Wales", "England", 1.06]

```

Figure 5.4: program output of pair-wise results for countries

In Figure 5.5 a larger matrix is applied to the program, here the graph displays the resulted pair-wise data of sites using a scatter plot graph in Microsoft Excel. The x-axis illustrate the sites in the collection, while the y-axis illustrates the similarity score. The legend on the right also depicts the sites which will be used to pair with the x-axis sites.

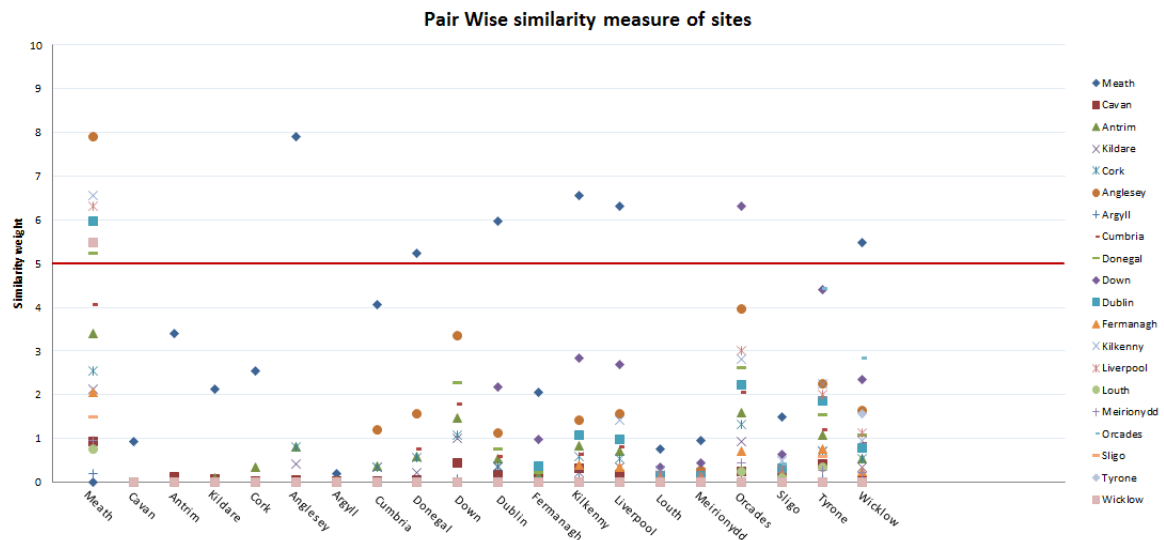


Figure 5.5: Graph representing pair-wise similarity measure of sites.

Looking at the graph above it is easily identifiable that the highest score at any point in the y-axis is 8 and lowest being 0, making 4 the average, keeping that in mind, a horizontal red line is drawn at 5. This highlights that any sites above that red line is highly similar to the site it's being paired to, and sites below the line is not as relevant.

For K-means clustering the *tombs* data is applied to the program as it contains the largest matrix compared to country and site. Looking at the visualized data below in figure 5.6, it contains groups generated by the K-Means program.

Here 4 centroids have been applied, generating 4 groups. The size of each orange bubble is generated(see Appendix B.7) by summing up the total weight generated by TF-IDF for each particular tomb, thus giving the bubble its size.

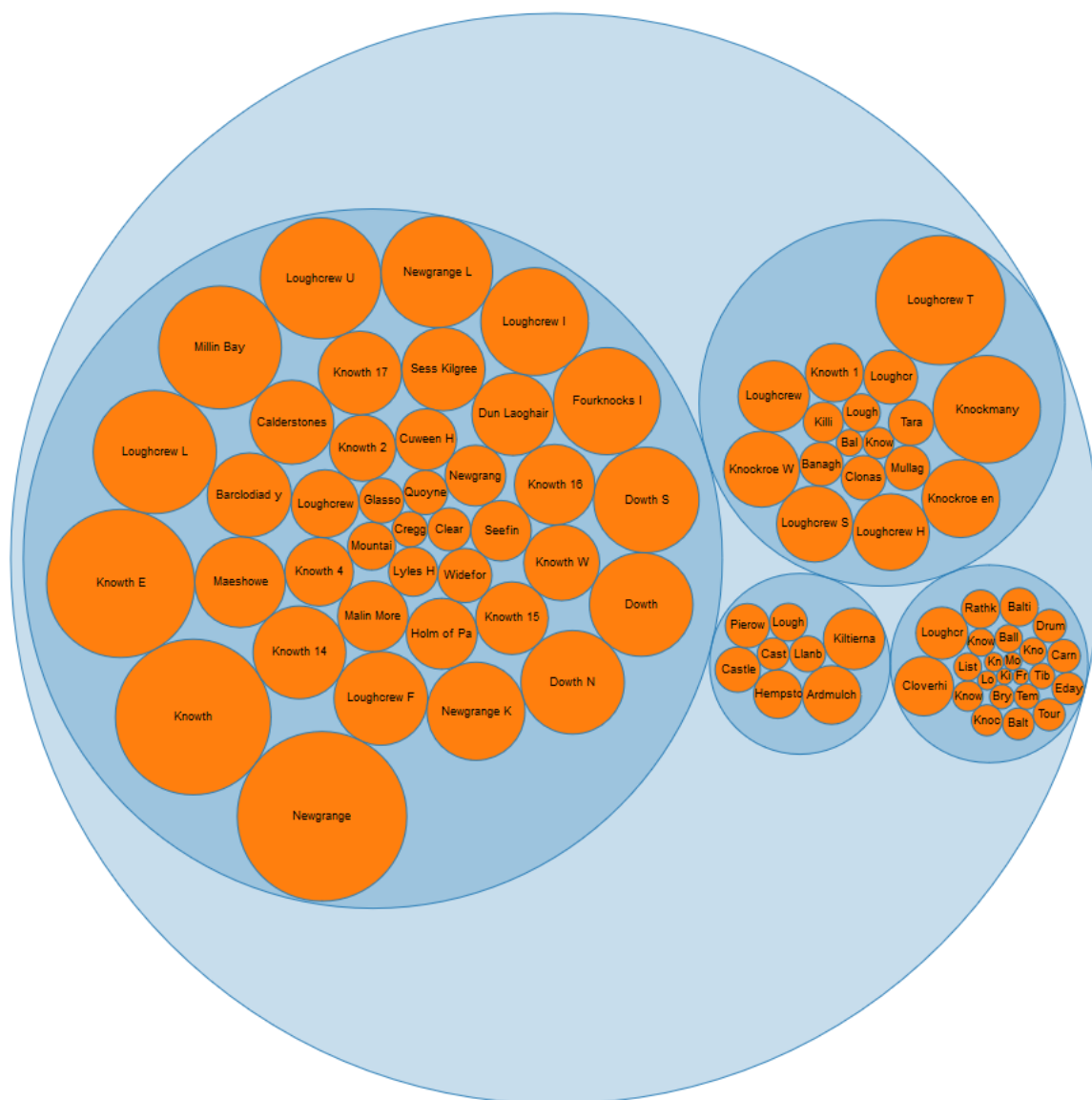


Figure 5.6: Graph illustrating the groups generated by the K-Means program

Chapter 6: Summary

This final chapter focuses on presenting the conclusions expanding from the background research to design and implementation, up to the summary of results. In the following sections is a recap of the work made that is used to establish conclusions of the project, critical analysis, lessons learned and future extensions to the project.

6.1 Critical analysis

While the project resulted in the main functionalities completed, along the way there were of course a number of various problems. Firstly prior to the implementation stage of the project, a detailed understanding of the VSM algorithms had to be acquired. This was achieved by creating the program step by step comparing the results generated by the computer by hand made calculations. To further understand the algorithm, different sources of the VSM algorithm was used and implemented while comparing each algorithm throughout the process. This highlighted parts of the code that was formulated wrongly and had to be re-implemented to give the right results.

Looking back at the input values and comparing it to the output results, an interesting finding was highlighted. The results show that the all the Irish entities(tomb and sites) are found highly similar to all other tombs in the Irish Sea, while the other tomb/site not in Ireland are found not quite similar to one another. Looking back at the input values, all the Irish tombs and sites seemed to have out numbered all other tombs greatly in terms of symbol frequency, even applying normalization techniques did not make much difference. In addition, the pair-wise algorithm should have ignored these unbalanced values.

6.2 Conclusion

This project was an interesting and educational experience which had given me a passion for the area of information retrieval and had given me a sense of the knowledge of a number of different similarity computation techniques. Initially the project goal was to develop a VSM program which will analyse symbols to find out any similarity between the different locations. This was achieved through a thorough investigation of the VSM technique and implementing a program that uses the technique. The resultant application runs smoothly and giving us exceptional results which illustrates that it is possible to use similarity algorithm to compare archaeological artefacts. This project provides a platform for future development and the project was in my opinion a success. The project was interesting and has given me a passion and insight that will see me continue to work in this area in the future.

6.3 Future work

This project has developed an application which could easily be expanded upon in the future. The VSM could be expanded to incorporate an automated image analysis; this program could take in numeric metric values from an image after converting that image into binary, these metric values taken can then be integrated as input instead of the symbol frequencies, however that type of application could use a more complex expansion of VSM. There are several models based on and expands the VSM such as; Generalized VSM, Rocchio classification and many others.

These models could also be used to calculate the accuracy of results which had been mention in the critical analysis (see section 6.1) by comparing the results given by this program with the results of one of the algorithm mentioned by using the same input values.

Bibliography

- [1] Wikipedia (2011). *Vector Space Model*. Available at: <http://en.wikipedia.org/wiki/Vectorspacemodel>. (Accessed 19 Nov 2013).
- [2] The eleventh International Word Wide Web Conference (2002). *Vector Space Model (VSM)*. Available at: <http://www2002.org/CDROM/refereed/643/node5.html> (Accessed 19th Nov 2013).
- [3] kjamby.kau.edu.sa. *K-means Clustering* Available at: Google: kjamby.kau.edu.sa k means clustering. (Accessed 3rd Dec 13)
- [4] Briard, J. *Les Megathithes de l'Europe atlantique. Architecture et art funeraire* (5000-2000 avant J-C) Paris: Errance.
- [5] Shee, E. 1973a *The techniques of Irish passage grave*. In : Kjaerum, P and Daniel, G.E (eds). *Megalithic Graves and Ritual : papers presented at the III Atlantic colloquium Moesgard*, . 1969. Kobenhavn : Jutland Archaeological Society publications 11, 163-72.
- [6] O'Sullivan, M. 1981b *Review of 'The Megalithic Art of Western Europe'*. by E. Shee Twohig. JRSOI 111, 127-30
- [7] Robin, G. (2008). *Neolithic passage tomb art around the Irish Sea Iconography and spatial organisation*. Volume 1. Unpublished PhD thesis. Universit De Nantes.
- [8] PlanetQuest (2005). *The History of Astronomy - Newgrange* Available at: <http://www.planetquest.org/learn/newgrange.html> (Accessed 20th Nov 2013).
- [9] E. Grogan (1991). *Prehistoric and Early Historic Cultural Change at Brugh na Binne*. Proceedings of the Royal Irish Academy. 91C. pp. 126-132.
- [10] Robin G. (2009). *L architecture des signes: L art parietal des tombeaux neolithiques autour de la mer d Irlande*. Available at: <http://www.academia.edu/241013/> 2009 Larchitecture des signes. Lar parietal des tombeaux neolithiques autour de la mer dIrlande.
- [11] Foucault, M (1966). *Les mots choses: une archeologie des sciences humaines*. Paris: Gallimard, page. 400.
- [12] Manning, C. D., Raghavan, P., and Schutze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.
- [13] Wikipedia (2013). *Ruby Programming Language*. Available at: [http://en.wikipedia.org/wiki/Ruby\(programming_language\)](http://en.wikipedia.org/wiki/Ruby(programming_language)) (Accessed 25th Nov 13).
- [14] Wikipedia (2013). *Neolithic*. Available at: <http://en.wikipedia.org/wiki/Neolithic> (Accessed: 2nd Dec 13).
- [15] World Heritage Ireland. *Education Pack for Junior Cert History*. Available at: http://www.worldheritageireland.ie/fileadmin/user_upload/documents/Edpackjr_1.pdf. (Accessed: 20th Nov 13).
- [16] Institute of Mathematics (2007) *Ruby User's Guide*. Available at: <http://www.math.bas.bg/bantchev/place/ruby-ug.html> (Accessed: 20th Nov 13).

- [17] Pantel, P., and Lin, D. (2002a). *Discovering word senses from text. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, . pp. 613-619, Edmonton, Canada.
- [18] Turney, P. D., and Littman, M. L. (2003). *Measuring praise and criticism: Inference of semantic orientation from association. ACM Transactions on Information Systems*,. 21 (4), p 315-346.
- [19] Rapp, R (2003). *Word sense discovery based on sense descriptor dissimilarity. In Proceedings of the ninth Machine Translation Summit* . pp. 315-322
- [20] Harris, Z. (1954). *Distributional structure. Word* . p 146-162.
- [21] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). *Indexing by latent semantic analysis. Journal of the American Society for Information Science* . (JASIS), 41 (6), 391-407.
- [22] Sokal, R.R., Sneath, P. H. (1963): *Principles of numerical taxonomy*. . Freeman, San Francisco & London
- [23] Lerman, I.C. (1970): *Les bases de la classification automatique*. . Gauthier-Villars, Paris
- [24] Jardine, N., Sibson, R. (1971): *Mathematical taxonomy* . Wiley, New York
- [25] Anderberg, M.R. (1973): *Cluster analysis for applications. Academic Press* . New York.

Appendix A

Appendix A.1














	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
	Abbréviat	Pays	Comté	Site	Dalle isolée	Dalle d'enceinte	Orthostat coulé	Orthostat chamé	Couverture	Pt. typique													
1	Am.1	IRL	Meath	Ardmulch	1					x	x		x					x		x	x		
2	Am.2	IRL	Meath	Ardmulch	2					x													
3	Bv.4	IRL	Meath	Ballinvally	dalle					x	x		x										
4	Bg.F	IRL	Meath	Baltinglass		F					x		x										
5	Bg.G	IRL	Meath	Baltinglass		G					x		x										
6	Bg.H	IRL	Meath	Baltinglass		H						x											
7	Bg.I	IRL	Meath	Baltinglass	J															x			
8	Bg.II.K	IRL	Meath	Baltinglass				K			x	x											
9	Bg.II.L	IRL	Meath	Baltinglass				L			x	x											
10	Bh	IRL	Cavan	Banagher	dalle						x							?		x	x		
11	Cm.Co1	IRL	Antrim	Carnanmo					Co1		x			x								x	
12	Cm.Co2	IRL	Antrim	Carnanmo					Co2	x													
13	Cd	IRL	Kildare	Castleder	dalle													x					
14	CI	IRL	Cork	Clear	dalle							x					x					x	
15	Cn	IRL	Meath	Clonasilla	dalle					x	x		x						?		x		
16	Ch.loose	IRL	Sligo	Cloverhill	loose					x	x		x										
17	Ch.C3	IRL	Sligo	Cloverhill				C3				x							x				
18	Ch.C8	IRL	Sligo	Cloverhill				C8			x	x							x				
19	Ch.C9	IRL	Sligo	Cloverhill				C9		x	x	x	x						x				
20	Cg	IRL	Meath	Cregg	dalle																		
21	Dh.K1	IRL	Meath	Dowth		K1					x	x	x						x				
22	Dh.K2	IRL	Meath	Dowth		K2				x													
23	Dh.K7	IRL	Meath	Dowth		K7				x		x											
24	Dh.K12	IRL	Meath	Dowth		K12					x	x	x										
25	Dh.K13	IRL	Meath	Dowth		K13					x												
26	Dh.K14	IRL	Meath	Dowth		K14						x											
27	Dh.K16	IRL	Meath	Dowth		K16					x		x								x		
28	Dh.K17	IRL	Meath	Dowth		K17					x	x	x								x		
29	Dh.K34	IRL	Meath	Dowth		K34							x										
30	Dh.K36	IRL	Meath	Dowth		K36																x	
31	Dh.K50b	IRL	Meath	Dowth		K50b																x	
32	Dh.K51	IRL	Meath	Dowth		K51				x	x		x	x					x	x	x		
33	Dh.K51b	IRL	Meath	Dowth		K51b					x								x				
34	Dh.K52	IRL	Meath	Dowth		K52					x												
35	Dh.K53	IRL	Meath	Dowth		K53					x												
36	Dh.K88	IRL	Meath	Dowth		K88					x		x	x									
37	Dh.N.L4	IRL	Meath	Dowth N				L4			x		x										
38	Dh.N.L5	IRL	Meath	Dowth N				L5			x									x	x		
39																							

Figure 6.1: Screenshot of Guillaume Robin’s database of symbols

Appendix A.2

This section illustrates a section of code snippet from the `frequency` model class

```
def self.read_file
  @csv_data = CSV.read('data.csv')
end

def self.country
  array_of_countries= ['Ireland','Scotland','Wales','England']
  country_frequency = []
end
```

```

array_of_countries.each { country_frequency << Array.new(13,0)}

@csv_data.each do |row|
  for i in 0..array_of_countries.count-1
    if row[2] == array_of_countries[i]
      for k in 3..14
        if row[k] == '1' || row[k] == '?'
          country_frequency[i][0] = array_of_countries[i]
          country_frequency[i][k-2] += 1
        end
      end
    end
  end
end
country_frequency
end

```

Appendix B

This appendix contains sections of code from the `vector space mode` class

Appendix B.1

```
def self.sum_document_freq
  @doc_freq = Array.new(13,0)
  @term_frequencies.each do |col|
    for i in 0..col.count-1
      if col[i] != 0
        @doc_freq[i] += 1
      end
    end
  end
  @doc_freq
end
```

Appendix B.2

```
def self.inverse_document_frequency
  output = []
  @doc_freq.each do |col|
    df = col.to_f
    score = Math.log10(N/df)      # formula
    output << score.round(3)
  end
  output
end
```

Appendix B.3

```
def self.term_frequency
  output = []
  @term_frequencies.each do |col|
    array = []
    array << col[0]
  end
end
```

```

    for i in 1..col.count-1
      tf = col[i].to_f + 1.0
      score = tf
      array << Math.log(score).round(3)
    end
    output << array
  end
p output
end

```

Appendix B.4

```

def self.tf_idf(tf, idf)
  output = []
  tf.each do |col|
    array = []
    k_means_array = []
    array << col[0] # adds name of place at start of array
    for i in 1..col.count-1
      i_d_f = idf[i-1].to_f # iterate through IDF
      tf = col[i].to_f # Iterate through TF
      score = i_d_f * tf # TF * IDF
      array << score.round(2) # to -> Pair wise similarity
    end
    output << array
  end
  output
end

```

Appendix B.5

```

def self.pair_wise_similarity(tf_idf)
  output = []
  tf_idf.combination(2) do |first, second| # e.g => [1,2,3,4] => [1,2][1,3][1,4]
    pairs = []
    multiplied = []
    pairs << first[0] # add pair names at start of array
    pairs << second[0] # add pair names at start of array
    for i in 1..first.count-1
      multiplied << first[i] * second[i] # multiply each pair values together
    end
    pairs << (multiplied.inject {|sum, n| sum + n}).round(2) # gives pair scores
    output << pairs # output all pair values
  end
  output
end

```



```

    end
end

```

Appendix B.6

```

def self.tf_idf_to_kmeans(tf, idf)
  output = []
  tf.each do |col|
    array = []
    k_means_array = []
    array << col[0] # adds name of place at start of array
    for i in 1..col.count-1
      i_d_f = idf[i-1].to_f # iterate through IDF
      tf = col[i].to_f # Iterate through TF
      score = i_d_f * tf # TF * IDF
      array << (100*score).round
    end
    output << array
  end
  output
end

```

Appendix B.7

```

def self.doc_length(tf_idf)
  output = []
  tf_idf.each do |row|
    add= []
    add << row[0]
    array = []
    for i in 1..row.count-1
      array << row[i]
    end
    add << (array.inject {|sum, n| sum + n})*100
    output<<add
  end
  output.each do |row|
    p row
  end
end

```

Appendix C

This appendix contains sections of code from `k means model` class

Appendix C.1

```
def self.hash(tf_idf)
  @hash = {}
  tf_idf.each do |col|
    hash2 = {}
    arr = %w{ dots_cupmarks circular spirals arcs radiate_circular
    semicircular chevrons triangles quadrangular scalariform straight waves} # keys

    (1...col.size).each { |d| hash2[arr[d-1]] = col[d] } # create a hash of hashes
    @hash[col[0]] = hash2 # add values
  end
  p '-'*100
  p @hash
end
```

Appendix C.2

```
def self.k_means(hash)
  for i in 0..10
    vector = hash
    @result = Kmeans::Cluster.new(vector, {
      :centroids => 4,
      :loop_max => 100
    })
    # The results differ for each run
    @result.make_cluster
    p @result.cluster.values
  end
end
```