



University of Thessaly
Department of Electrical & Computer Engineering

Postgraduate Thesis

**Determination of user's social and psychological profile
based on algorithmic and statistical analysis**

Anastasios Alexandridis

**Supervisor professor
Michael Vasilakopoulos**

**June 2017
Volos**

University of Thessaly
Department of Electrical & Computer Engineering

Postgraduate Thesis

**Determination of user's social and psychological profile
based on algorithmic and statistical analysis**

Anastasios Alexandridis

**Michael
Vassilakopoulos**

**Panagiota
Tsompanopoulou**

**Panayiotis
Bozanis**

**June 2017
Volos**

Πανεπιστήμιο Θεσσαλίας
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Μεταπτυχιακή Διατριβή

**Καθορισμός κοινωνικού και ψυχολογικού προφίλ χρήστη
βασισμένο στην αλγοριθμική και στατιστική ανάλυση**

Αλεξανδρίδης Αναστάσιος

**Μιχαήλ
Βασιλακόπουλος**

**Τσομπανοπούλου
Παναγιώτα**

**Μποζάνης
Παναγιώτης**

Ιούνιος 2017
Βόλος

This work is dedicated, as a small token of appreciation, not only to the people who give their life in order to understand the human nature analyzing it by using scientific methods, but also to the people who build these incredible computational machines.

Abstract

The human behavior algorithms have been the holy grail for a lot of researchers from different fields such as computer science and computational psychology. The usual research subjects are related to the emotional detection & classification, social behavior and the Natural language Processing -NLP-, and the researchers work has been focused on the performance and accuracy, presented by the machine learning algorithms which are used to address the above research subjects, but little has been done for the quality of the data which is relevant to the human behavioral algorithms and the structural representation of the data.

This thesis approaches the subject of computational psychology in a different way and focuses on the reduction of bias error via a strategy for data mining which guarantees more structural and robust data. As it is known, bias error is responsible for shifting the center of the target from reality to an unreal situation, which affects the quality components of the data as well as the cycle of the data quality. The project constitutes a computer based method for the classification of documents to form graphs, which are the final product and drawing-up of the users psychological and social profile. Also, it detects the most important nodes which are the main themes of a subject's life.

This method was based on the Self Memory System -SMS- which was proposed by the Conway and Playdell Pearce in 2000. Our model follows exactly the graphical representation of the conceptual model of Conway and Playdell Pearce which is used as a blueprint for the mining of data by the social media (e.g. facebook), and the formation of the "Autobiographical Knowledge Base" in a structural way, which can be processed by the computational devices. Moreover, according to Conway's model the autobiographical memory is formed by episodic memories which are drawing the events of a theme in a graph. Finally the graph's most important node is the representative event which indicates the theme.

Techniques such as the MaxEntropy, Logistic Regression, and Naive Bayes are used extensively, and tested for their accuracy and their efficiency, by using the most accepted methods like the Receiver Operating Characteristic -ROC- curve. As concerns the relation between different events, which is vital for the evaluation of the nodes and the formation of the edges, a verified

technique is used which is known as cosine similarity. Last but not least, one of the most recently developed social analysis graph methods, degree centrality is used for the determination of the most central nodes. The level of centrality indicates the importance of a node as concerns the representation of the theme.

Key words: human behavior algorithms, computational psychology, emotional detection & classification, social behavior, Natural Language Processing, machine learning algorithms, bias error, data mining, cycle of data quality, Self Memory System, conceptual model, social media, Autobiographical knowledge base, MaxEntropy, Logistic Regression, Naive Bayes, Receiver Operating Characteristic curve, cosine similarity, degree centrality.

Volos 2017

Περίληψη

Η συγκεκριμένη μεταπτυχιακή διατριβή πραγματεύεται το θέμα των αλγορίθμων της ανθρώπινης συμπεριφοράς, το οποίο, όπως είναι γνωστό, απασχολεί πολλούς ερευνητές της επιστήμης των υπολογιστών και της υπολογιστικής ψυχολογίας. Ωστόσο θα πρέπει να επισημανθεί ότι η εν λόγω ερευνητική εργασία διαφοροποιείται από τις μέχρι τώρα ερευνητικές μελέτες, οι οποίες εστιάζουν το ενδιαφέρον και την προσοχή τους στην ακρίβεια των αλγορίθμων μηχανικής εκμάθησης του εντοπισμού και της κατηγοριοποίησης των συναισθημάτων, της κοινωνικής συμπεριφοράς και της επεξεργασίας των φυσικών γλωσσών, που κατά γενική ομολογία αποτελούν κοινότοπη θεματολογία. Σε αντίθεση λοιπόν με τις μελέτες αυτές το επίκεντρο αυτής της διατριβής εστιάζεται στην διερεύνηση της ποιότητας και της δομημένης αναπαράστασης των δεδομένων, το οποία είναι σχετικά με τους αλγορίθμους της ανθρώπινης συμπεριφοράς.

Ειδικότερα η διαφοροποίηση της έγκειται στο γεγονός ότι η προσέγγιση του θέματος επικεντρώνεται στην μείωση του συστηματικού σφάλματος το οποίο είναι υπεύθυνο για την ολίσθηση του αντικειμενικού στόχου από μία πραγματική σε μία φανταστική κατάσταση, επηρεάζοντας τους ποιοτικούς δείκτες δεδομένων καθώς και τον κύκλο ποιότητας. Μάλιστα στην εν λόγω εργασία προτείνεται μια υπολογιστική μέθοδος για την κατηγοριοποίηση εγγραφών, έτσι ώστε να είναι εφικτός ο σχηματισμός γράφων, οι οποίοι αναπαριστούν το κοινωνικό και ψυχολογικό προφίλ του χρήστη αλλά και για τον εντοπισμό των πιο σημαντικών κόμβων οι οποίοι είναι τα κύρια θέματα της ζωής ενός αντικειμένου.

Αυτή η μέθοδος βασίζεται στο μοντέλο Self Memory System -SMS-, το οποίο προτάθηκε από τον Conway και Playdell Pearce το 2000 και επιλέχθηκε ως πρότυπο στη διατριβή αυτή, διότι έχει την δυνατότητα συμπαγούς και ακριβούς αναπαράστασης δεδομένων. Πιο συγκεκριμένα η παραπάνω μέθοδος ακολουθεί ακριβώς αφενός την γραφική αναπαράσταση που χρησιμοποιείται ως πρότυπο για την εξόρυξη των δεδομένων από τα μέσα κοινωνικής δικτύωσης (π.χ. Facebook), και αφετέρου τη διαμόρφωση της βάσης γνώσεων της Αυτοβιογραφικής Μνήμης με έναν δομημένο τρόπο ο οποίος είναι επεξεργάσιμος από τις υπολογιστικές μηχανές.

Για τη μορφοποίηση της παραπάνω μεθόδου επιλέχθηκαν και έγκυρες τεχνικές, όπως η Μέγιστη Εντροπία, η Λογιστική παλινδρόμηση και η αφελής εκδοχή του κανόνα Bayes, οι οποίες

χρησιμοποιούνται εκτεταμένα και έχουν δοκιμαστεί για την ακρίβεια και την αποτελεσματικότητα τους, χρησιμοποιώντας τις πιο αποδεκτές τεχνικές ελέγχου, όπως η Receiver Operating Characteristic -ROC- curve. Όσον αφορά τη σχέση μεταξύ διαφορετικών γεγονότων ζωτικής σημασίας για την εκτίμηση των κόμβων και την διαμόρφωση των γράφων επίσης χρησιμοποιείται μια δοκιμασμένη τεχνική, η οποία είναι γνωστή ως συνημιτονοειδής ομοιότητα, ενώ για τον καθορισμό των πιο κεντρικών κόμβων χρησιμοποιείται μία εξίσου έγκυρη τεχνική, η degree centrality. Τέλος το επίπεδο της συγκεντροποίησης υποδεικνύει την σημασία ενός γράφου όσον αφορά την αναπαράσταση του θέματος.

Λέξεις κλειδιά: Αλγόριθμοι ανθρώπινης συμπεριφοράς, υπολογιστική ψυχολογία, εντοπισμός και κατηγοριοποίηση συναισθημάτων, κοινωνική συμπεριφορά, επεξεργασία φυσικής γλώσσας, αλγόριθμοι μηχανικής εκμάθησης, συστηματικό σφάλμα, εξόρυξη δεδομένων, κύκλος ποιότητας δεδομένων, Self Memory System -SMS-, μέσα κοινωνικής δικτύωσης, Βάση Γνώσης Αυτοβιογραφικής Μνήμης, Μέγιστη Εντροπία, Λογιστική Παλινδρόμηση, Αφελής εκδοχή του κανόνα Bayes, συνημιτονοειδής ομοιότητα, degree centrality.

Βόλος 2017

Contents

1	Introduction	15
1.1	The scope of this thesis	15
1.2	Behavior Algorithms	16
1.3	Computational Psychology	17
1.4	The Autobiographical Memory	17
1.5	Related work	18
1.6	Structure of thesis	21
2	The description of the computational method of Conway's Hierarchical Model	23
2.1	Description of the sequence like the formation of Conway's Model	23
2.2	The Event Specific Knowledge	25
2.2.1	Extracting info from the ESK's corpus	25
2.3	The General Events	26
2.3.1	Forming the graphs based on events	26
2.4	Lifetime Periods	27
2.4.1	Extracting the themes from the graphs	27
3	The technical aspects of the Python and Jupyter notebooks	29
3.1	Description of Python	29
3.2	The scientific version of Python (Anaconda)	30
3.2.1	Scipy	30
3.2.1.1	Numpy	31
3.2.1.2	Scipy library	31
3.2.1.3	Matplotlib	31
3.2.1.4	Pandas	32
3.2.1.5	Sympy	32
3.2.1.6	nose	32
3.2.1.7	Ipython	33
3.2.1.8	Scikit Learn	33
3.3	The Jupyter Notebook	34
3.4	Graphviz and NetworkX	34
3.5	Installation process of Python and Jupyter notebook	35
3.5.1	Installation of the Python Anaconda 2.7 version	35

3.5.2	Starting the jupyter notebook	37
3.5.3	Install Graphviz and NetworkX	39
4	The technical aspects of the NLP Core	41
4.1	Description of the Natural Language processing Core	41
4.2	Encapsulation of the NLP core to Python using Json	42
4.2.1	The Json	42
4.2.2	Installation process of NLP core	42
4.2.3	Commands for encapsulation of NLP core using Json	43
5	The development of the computational method and experiments	45
5.1	The general overview of the computational method	45
5.2	The data sets (Importing data)	46
5.2.1	Privacy policy	46
5.2.2	The data files	47
5.2.3	Import data	47
5.3	Vectorization of the data	49
5.3.1	Unigram language model	49
5.3.2	Sparsity	51
5.4	The Multinomial Naive Bayes	51
5.5	Confusion error for Naive Bayes	52
5.6	Evaluation of Multinomial Naive Bayes using the ROC Curve	53
5.7	Logistic Regression	54
5.8	Evaluation of Logistic Regression using the ROC curve	56
5.9	Frequency Ratio	56
5.9.1	Token counting	56
5.9.2	Token Frequency & ratio	58
5.9.3	The commonness between the tokens	59
5.10	The Cosine Similarity (forming the graphs)	60
5.11	The centrality and the network X (forming the themes)	65
5.11.1	The types of centrality	65
5.11.2	The degree centrality	65
6	Conclusion & future Plans	75
6.1	Conclusion	75
6.2	Future Plans	76
7	Bibliography, references, and links	77

1. Introduction

The introduction is aiming to give the fundamental information about the theory as well as an intuitive presentation of the scope and the structure of this thesis. Also the related work is mentioned with specific references, which is an important section, because it gives an holistic view of the subject and shows the originality of this piece of work.

1.1 The scope of this thesis

The scope of this thesis is the modeling and the development of a computational method that determines the user's psychological and social profile based on Conway's hierarchical model (see Figure 2) which forms the structural representation of the life story of a subject (human). By the term life story we mean the information of the autobiographical memory of a subject (see: 1.4 The Autobiographical memory) which also exists in a more robust and unspoiled form inside to the text raw material of social media.

As becomes obvious from the above, the input is raw text material from posts in social media sides.

As concerns the originality, according to the best of our knowledge, this piece of work is unique in a global scale. There is some similar piece of work as concerns the autobiographical memory (see related work) but none has focused on the text material of the social media. Although there is a lot of work about the text classification, as well as sentiment detection, it seems that little has been done to representing or determine the social and psychological factors in a computational form, based on the classified material of these sources.

To achieve its goal, a variety of supervised learning theories on text classification and statistical modeling of text has been used extensively. As concerns the development of the method in a practical and applied form, a combination of tools has been used for the development and production of the necessary code. These tools are:

- **Anaconda Python**, is the scientific version of the Python.
- **Scikit Learn**, an included tool kit in python Anaconda which has been specifically designed for machine learning algorithms with Python.
- **Graphviz**, A tool for modeling the formation of the graphs.
- **Network X**, a tool for developing graph analysis methods in Python.
- **Matplotlib**, a tool for the visualization of graphical presentations.
- **Json**, a special tool for including Java projects and Java code in python. It also works with other languages such as C, C#, Perl.
- **Jupyter notebook**, a web tool for the presentation and development of code for a variety of Programming Languages such as Python.
- **LibreOffice Calc**, a spreadsheet tool of the LibreOffice suit which is used for manual pre-processing and automatically importing the data sets in a readable form for the Python and the Scikit Learn.

1.2 Behavior Algorithms

Human behavior algorithms are an essential part of the “human computing” which, as a term, is interpreted as the automatic sensing and understanding of human behavior in computer-supported and smart environments [1].

The human behavior is detected and analyzed according to data which have been extracted by face and body language recognition models [1], voice recognition models, and natural language textual input [2]. As concerns the goals of the analysis, they are aiming to the analysis of sentiments and autobiographical knowledge.

Moreover, according to Mao Quiushi a good model for human behavior “is not only simply evaluating a particular system; a broader objective for studying behavior is to develop generalized models that apply in different contexts [3- page 22]

Having understood the human behavior algorithms, we realize that an important source of Big Data which are related to a subject is the social media.

1.3 Computational Psychology

According to Ron Sun, Computational psychology or computational cognitive modeling “explores the essence of cognition (including motivation, emotion, perception e.t.c.) and various cognitive functionalities through developing detailed, process-based understanding by specific corresponding computational models” [4-page 3]. Moreover, we deal with 3 categories of models mathematical, computational, and verbal conceptual models.

The computational models are algorithmic processes. The mathematics are presented by mathematical equations and the entities, relations, and processes are presented by Verbal-conceptual models which are using natural languages [4].

In our case the modeling of the autobiographical Knowledge base which is based on the Conway’s structural representation is a mixed model. First, it is an algorithmic process for 2 reasons a) a well structural representation by Conway b) serial algorithmic sequence of the main steps which are formed by major algorithms such as Naive Bayes and degree centrality. Second, it is a mathematical process because it uses arithmetic representation for the input data (vectorization of the data), to fit variables of well defined statistic methods such as Multinomial Naive Bayes. Last, it is a Verbal- Conceptual model because uses text information as data sets and training prototypes which are informal natural languages.

1.4 The Autobiographical Memory

Definition: *The autobiographical memory is a long-term memory which is formed by episodic (specific episodes) and semantic memory (conceptual, generic, schematic knowledge of our lives) [5].*

The Long Term Memory -LTM- is anything that can be remain for 30 sec. The LTM formed from Declarative and Procedural which are explicit and implicit respectively. Moreover the Declarative is formed by the episodic and semantic (Figure 1) [5][6]. The long-term memory contains knowledge and episodic memories related to long-term goal processing (e.g. completing a work project) [7].

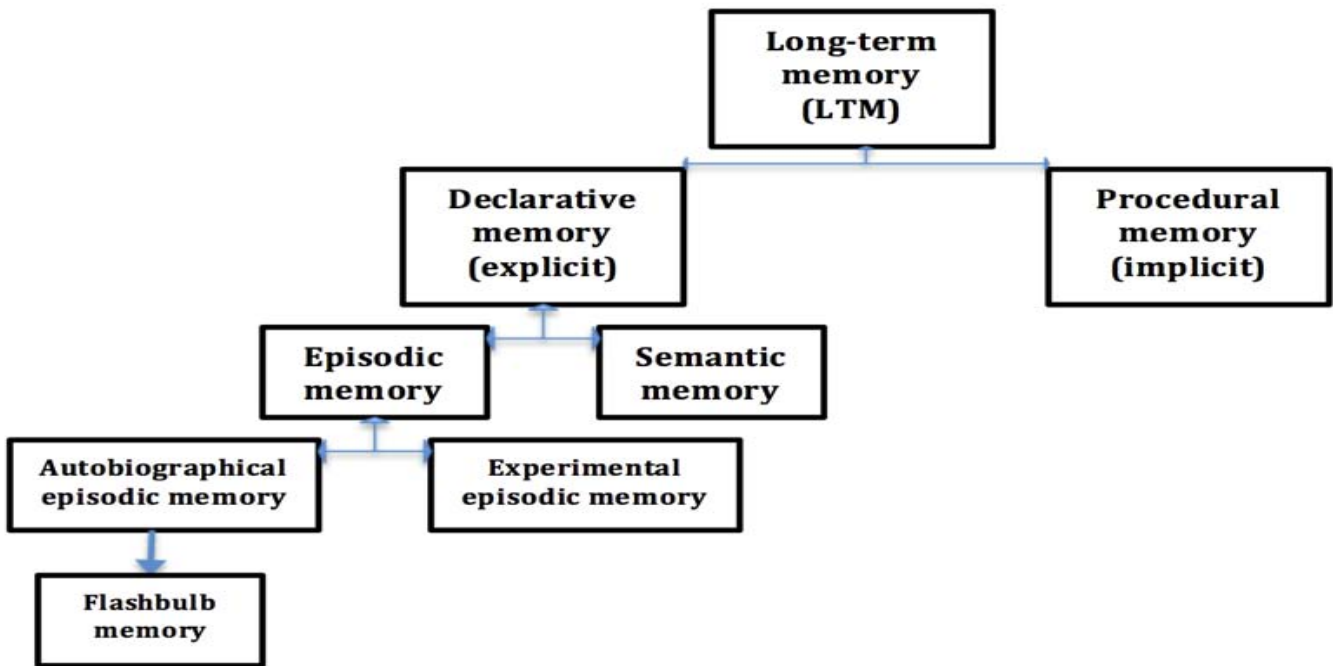


Figure 1 - The Long-Term Memory [5]

Episodic memory

Definition: According to [6] is the memory for “events that occurred in particular spatial and temporal contexts”, and is a type of declarative memory.

Semantic memory

Definition: According to [6] “ is the capacity for recollecting facts and general knowledge about the world”, and also is a type of declarative memory.

1.5 Related work

This section starts with the general related work about the human behavior algorithms and the computational psychology. In a second phase, the presentation focuses on the work as concerns text classification. Finally, in the end of the section the related work for the autobiographical memory is presented.

Starting with computational psychology, an important part is the sentimental analysis. One of the most important works is this of Xinzhi Wang, Xiang feng Luo, Jinjun chen, with title “Social Sentiment Detection of Event via a Microblog”[8]. They are forming a sentiment vector for the extraction of a sentiment distribution of a blogger on event, and they are using this sentiment vector combined with a general social sentiment model, to compute the social state. In more detail, this work uses a search engine for the collection of data relative to a specific topic. Then, the data collection is used for the identification of emotions (joy, anger, sad, fear, surprise, disgust) which are based on the training set. Additionally, by using Pointwise Mutual Information -PMI- their model is able to discover the relation between the identified emotions.

A piece of work that combines the sentiment and text classification is that one of Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. This specific study gives a new perspective by classifying text documents not according to topic, but by positive and negative sentiment. The team used and tested 3 famous machine learning algorithms (Naive Bayes, Maximum Entropy, and support vector machines). Moreover, they discovered that the algorithms do not perform as well in sentiment classification as they do on topic-based categorization [9].

Another very important method for text classification has been presented by Andrew Kachites NcCallum and Karnal Nigam, and is known as “pool-based active learning”. This method is labeling a large corpus of unlabeled documents, by using density weight pool based sampling and Expectation Maximization -EM-. Moreover, the training data is sparse, and by using the Query by Committe -QBC- algorithm, the selection labeling from the pool with the unlabeled documents is possible. The combination of EM is used for the co-occurred vocabulary in unlabeled documents of the pool. The study has shown that this method reduces the necessity for labeled examples significantly [10].

Also another work of the two prementioned researchers, and John Lafferty, uses maximum entropy techniques for text classification. To “heal the wound” of the overfitting the paper proposes a Gaussian Prior. Also the importance of the features for the maximum Entropy is clearly presented, and the size of the vocabulary plays an essential role to the size of the error as concerns inaccuracy [11].

As concerns other projects which focus on text classification, we can mention that of Andrew Mcallum and Lamal Nigam, which is comparing the Naive Bayes Multivariable Bernouli model and the multinomial model. Their conclusion is that the multinomial model is better in that it reduce the average error from 27% to 59% [12].

A more detailed and wider piece of work about the performance in text classification of Naive Bayes different versions is this of Vangelis Metsis, Ion Androutsopoulos, Georgios Paliouras.

This classification is focusing on spam filtering but this has not any negative effect on the performance of text classification like in all other cases. For the evaluation of data, roc curves are used. Also two not so widely used Naive Bayes methods are evaluated, which are Flexible naive Bayes and the Multinomial with Boolean Attributes. According to the result both perform very well [13].

In their work, Andrew Kope, Caroline Rose, and Michel Katschabow present a hierarchical network model which simulates human autobiographical memory. The model is agent based and consists of three layers known as Immediate, short, and long term pool, which encapsulate a given memory pattern. These patterns are represented as a collection of interconnected nodes, and are composed of six components (ID, Keywords, Type, Emotional Valence, weight, and Timestamps). The whole process is controlled by a manager which performs the functions of handling the integration of new events, activating nodes within the networks, and simulating the decay of memories over time. The behavior of the agent is dictated by reactive and rational processing units which are based on the psychological state factors such as personality traits emotions/mood, and social ties with other agents. Finally, the model has been implemented to the video game known as Minecraft by using the above mentioned agents which represent non-playable characters -NPC- [14].

The last work which shares a common psychological theory with this thesis is the model of Di Wang, Ah-Hwee Tan, Chunyan Miao, and is known as AM-ART which stands for Autobiographical Memory- Adaptive Resonance Theory Network. The main goal of the paper is different than the one this current thesis has, and aims to “capture autobiographical memories, comprising pictorial snapshots of one’s life experiences together with the associated context, namely time, location, people, activity and emotion”(as it is referenced in the abstract of the paper – Modeling Autobiographical Memory[14]). The paper shares the same theoretical basis with the current thesis as concerns the psychological theory of Conway’s hierarchical model for the levels of specificity [15], but with a very different approach and for a very different reason. The thesis uses the Conway’s hierarchical model as a blueprint for text classification. However, this paper has important characteristics which are very essential factors to be mentioned and referenced.

The AM-ART is based on the Adaptive Resonance Theory -ART- which is a self organized neural network [16]. The initial version of ART uses two layers [16] but the AM-ART version has 3 layers, and each one represents the hierarchy of Conway’s model (life-time periods, general events, and event-specific knowledge) [15-section 4]. The first layer which is responsible for the Event-Specific Knowledge collects information about time, location, people, activity, emotion, imagery, via “six input fields”. The second layer is responsible for the activation of “codes” and the weighing

of the input learning pattern of the first layer of the six fields. The process in layer 2 is repeated until a “winner code” is selected, if and only if it satisfies the “vigilance criteria” [14- section 4.1. and Algorithm 1]. Layer three contains the final product of the second layer which are the episodes.

According to international bibliography and publications the “Modeling Autobiographical..” is probably, the first work which simulates the phenomenon of wandering [14]. The wandering phenomenon is beyond the scope of this thesis.

1.6 Structure of thesis

This thesis has been separated in seven part or chapters. The first chapter with the title “Introduction” presents the scope of the thesis, concepts of general knowledge (e.g. Behavioral algorithms, computational psychology) which is vital for the understanding of this piece of work in a theoretical level.

In the second chapter, the computational method of Conway’s hierarchical model is presented from a descriptive and theoretical scope.

The third and the fourth chapters follow with the presentation of the tools which are used for the development of the computational method in an implemented form.

Next in the line is the fifth chapter with the development of the computational method in an applied form using the tools which have been listed above. A small example which has been designed to satisfy both, a very difficult case of data (very little raw text material, shortage of vocabulary e.t.c.) and a presentable formation for presenting the functionality of the method is developed step by step.

Last but not least, the conclusions about the method are given in an detailed form as a synopsis of all the project.

Finally, the last part of the thesis lists the bibliography, the links, and, generally speaking, the references of the thesis.

2. The description of the computational method of Conway's Hierarchical Model

This chapter presents the computational method of Conway's model from a descriptive and theoretical perspective. The graphical representation of Conway's model, as well as terms such as Event Specific Knowledge -ESK-, General Events and Life Time Periods are presented and described with specific references to Conway's work and the work of others.

2.1 Description of the sequence like the formation of Conway's model

Our aim is to follow and reproduce Conway's model [15] [17] in a web based system which uses information from the web (especially from the social media), relative to the "autobiographical knowledge base" of a subject (by the term subject we mean a human).

The input is raw text material from different posts of the subject. The text is classified in events based on the textual context of the posts. This is a goal-based process which follows the rules of supervised learning, so algorithms such as Naive Bayes and Logistic regression are used [9][10][11][12][18][19][13][20], based on training data which have been labeled by hand [18]. Moreover, the automated formation of new types of events is possible by using unsupervised learning such as clustering algorithms, but this is beyond the scope of this thesis.

The classified data into general events are checked for their similarity by using the cosine similarity[21][22] for vector space models [18]. This method is harmonized because our data are formed in vectors.

After the draw of edges based on the similarity [21][22] of the events we locate the central event by using the method of the degree centrality for the detection of communities in a graph of social media[23][24]. The combination of a community and a central node represents a theme. These are the themes of the life story of the subject according to Conway's schematic representation (Figure 2).

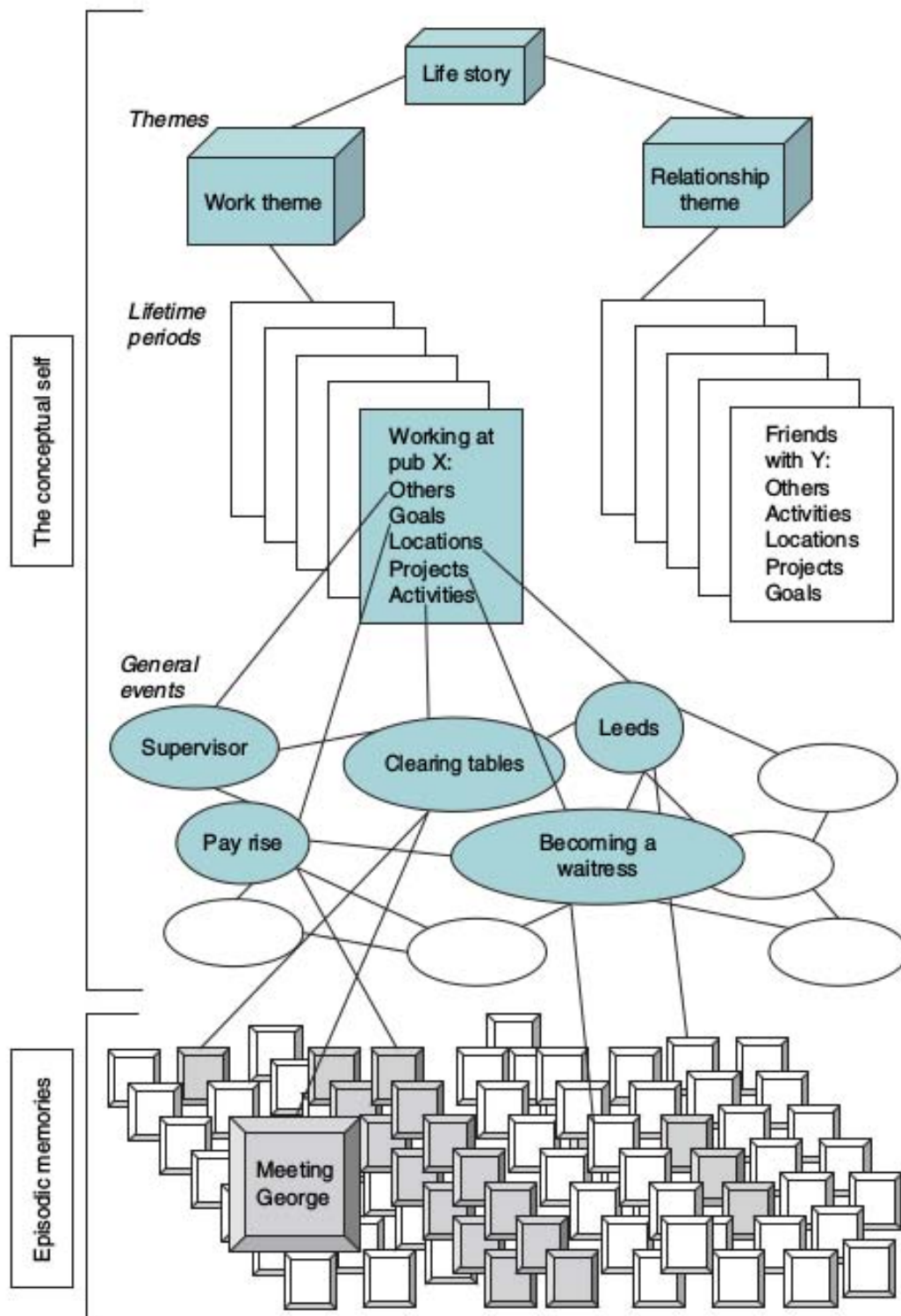


Figure 2 – The knowledge structures in autobiographical memory, source [17]

2.2 The Event specific knowledge

The event specific knowledge contains all the information about the subject's life. In our case, Event Specific Knowledge are small posts of the social media corpus which are related to the life story of a subject.

2.2.1 Extracting info from the ESK's corpus

The corpus of the ESK's lies on the social media corpus of different social media sources in text form. All the data sets are derived by the posts of the user's profile by using specific API tools relative to the data extraction from the source. The initial method which is used for the classification of the raw text material is the bag of words, which is most common of all for text classification, sentiment analysis, and spam detection modeling [9] [10] [11] [12] [18] [19] [13] [20]. Our method follows the same rules as the architecture of the above mentioned fields of applications which is the categorization of texts in specific classes. These classes are forming the general events which are going to form the graphs of each theme [15][17] (see Figure 2).

Sort texts such as social media posts, are limiting the performance of the text classification and the bag of words [25]. This thesis faced the same problem as the work of Bharath Sriram, David Fuhry, Engin Demir, Hakan Ferhatosmanoglu, Murat Demirbas [25] and acted similarly by predefining the generic classes such as (Marriage, Work, Holidays, Children, Studies e.t.c. - for more info see: 5.2 The data sets (Importing data)).

According to the technique of the bag of words each document (consisted of small sentences) is represented as a bag (multiset) of the contained words [20]. The data are transformed into numeric vectors based mainly on their frequency of occurrence in the text (see: 5.2 Vectorization of the data).

Note: The corpus of the social media may contain not only information about the episodic memories but also data for semantic information. Because of the use of predefined labels for the classes the semantic information is limited to an error which is a side-effect of the statistic model for classification.

2.3 The General Events

According to the hierarchical model [15] [17] the general events are nodes of a graph which have been extracted by the Event specific knowledge -ESK- and the contained information consists mainly of episodic memories. The node represents a class which contains specific information and has a predefined label. Each class contains different information (wife, family, children, colleagues, holidays e.t.c.). In this phase the cosine similarity is used to see how similar two events (nodes) are, by finding similar key words: e.g. the posts about the “family” label (event relative to family-node) will contain the common with the marriage label frequently words (children, wife, parents), and different words such as, best men (significantly larger frequency in marriage label), grandparent (significantly larger frequency in family label). Also these two sets have a different main concept (family and marriage). The common words between these two events form a strong relationship between them.

Because we have categorized the raw material the cosine similarity is an effective metric for finding the relationship between two classes (if they are similar or not). Moreover, to build a predefined lexicon with all the common words and train a model for the similarity would be very difficult, if we consider, for instance, the fact that the names and the places are significant information about the similarity and cannot be predefined. So, such an automated classification mechanism for the relationship is necessary.

2.3.1 Forming the graphs based on events

By checking the cosine similarity we check the main parts of the phrases (posts) for similar words inside to nodes and how similar these nodes are. If the similarity is high the system draws an edge between the node by saying that they contain similar words to the classified categories so these classes are relative to each-other. If the data was not classified or pre-labeled, the similarity would not be considered an effective metric. Also the filtering for common words of speech is pre-requested since words like, the, as, on, or, will cause a confusion and a bias result to the metric of cosine similarity indicating there is a relationship in almost every node (see: 5.10).

This stage, practically, is forming the graph of the general events of the autobiographical memory (Figure 2).

2.4 Lifetime Periods

The themes form the lifetime periods. Each graph represents a theme which, if we see it, is formed by the most important events.

2.4.1 Extracting the themes from the graphs

If we consider the fact that we have used the similarity to form the graph, a sub-graph of the life story graph is a theme. Also the most central node indicates the label of the theme (see Figure 2). This is verified by the common logic of the graph analysis. The central graph contains more information similar to all the other nodes of the sub-graph (theme). So, the detection of the sub-graph (theme) and its central node (each acts as a label) is possible by using the centrality from the graph theory [23][24]. For more information, see: 5.11 The centrality and the Network X (forming the themes).

3. The technical aspects of the Python and Jupyter notebooks

This chapter describes all the tools which are used for the development of the computational method for the Conway's model. Python Anaconda, Scikit Learn, Graphviz and NetworkX combined with the Jupyter notebook are the main tools. All the others are vital in the background processes in order to obtain the right functionality of the above mentioned tools.

3.1 Description of Python

Python is a high level and a general purpose language which was originally designed and introduced by Guido van Rossum on 20 February 1991, with the 0.9.0 version [26].

After more than 25 years of history and evolution Python took its final form which is the latest version known as 3.5. However, the 2.7 version is still very popular due to the support of a lot of packages such as NetworkX.

The language is a mixed model as concerns the styles, with object-oriented, imperative, functional programming and procedural abilities [27].

Moreover, it has a huge library with a lot of packages such as Scikit Learn which is used for machine learning by giant tech companies such as Google and Facebook [20].


The applications of Python are in many fields such as the web and Internet development, Database access, Desktop GUI's, Scientific and Numeric computation, Education, Network programming and finally software and game development [28].

As concerns the usage, Python is very human friendly and allows the user to achieve the same result as with languages like Java and C by using fewer lines of code and more effective programming paradigms. Moreover, Python does not exclude languages such as Java but encapsulates them using tools such as JavaScript Object Notation -Json-, a tool which is a data-interchange language for different programming languages like C, C#, Java, Python, Perl, and many other [29].

3.2 The scientific version of Python(Anaconda)


The scientific version of Python which is known as Anaconda, is a combination of the main language and packages for scientific and numeric computing. Python Anaconda mainly includes:

- Ipython
- Scipy
- Numpy
- Pandas
- Matplotlib
- Scikit Learn

Logo	Site
	https://www.continuum.io/

3.2.1. Scipy

The scipy is an open source stack collection and is composed of packages which are aiming to the scientific computing [30]. These packages are: Numpy, Scipy library, Matplotlib, Pandas, Sympy, nose, Ipython.

Logo	Site
	https://www.scipy.org/

3.2.1.1 Numpy


Numpy is an open source basic package which is combined with Python in order to conduct scientific computing. The main features of the tool are:

- A powerful N dimensional array object
- sophisticated (broadcasting) functions
- tools for integration
- useful linear algebra, Fourier transform, and random number capabilities

Logo	Site
	http://www.numpy.org/

3.2.1.2 Scipy library

It is an open source library of numerical algorithms for numerical intergreption and optimization.

Logo	Site
	https://www.scipy.org/

3.2.1.3 Matplotlib

Is an open source library that focuses on the plotting of figures for numerical and mathematical presentations.

Logo	Site
------	------



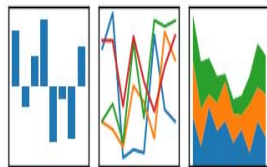
<http://matplotlib.org/#>

3.2.1.4 Pandas

Pandas is an open source library of tools for conducting data analysis with Python. Among to other things Panda is able to handle large data frames as arrays effectively.

Logo

pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Site

<http://pandas.pydata.org/>

3.2.1.5 Sympy

Sympy is an open source library which is used for symbolic mathematics.

Logo



Site

<http://www.sympy.org/en/index.html>

3.2.1.6 nose

It is an open source tool for conducting testing such as: parallel testing (Multiprocess), doctest, debug, code coverage tests.

Logo



Site

<http://nose.readthedocs.io/en/latest/>

3.2.1.7 Ipython

Ipython is an open source shell which is used for interactive computing with python.

Logo



Site

<https://ipython.org/>

3.2.1.8 Scikit Learn

Is a tool kit that is contained in Python Anaconda and is used for machine learning modeling. It contains tools for the development of methods by using techniques such as: Multinomial Naive Bayes, Logistic regression, Cosine Similarity, Lasso linear and other types of regression.

Logo

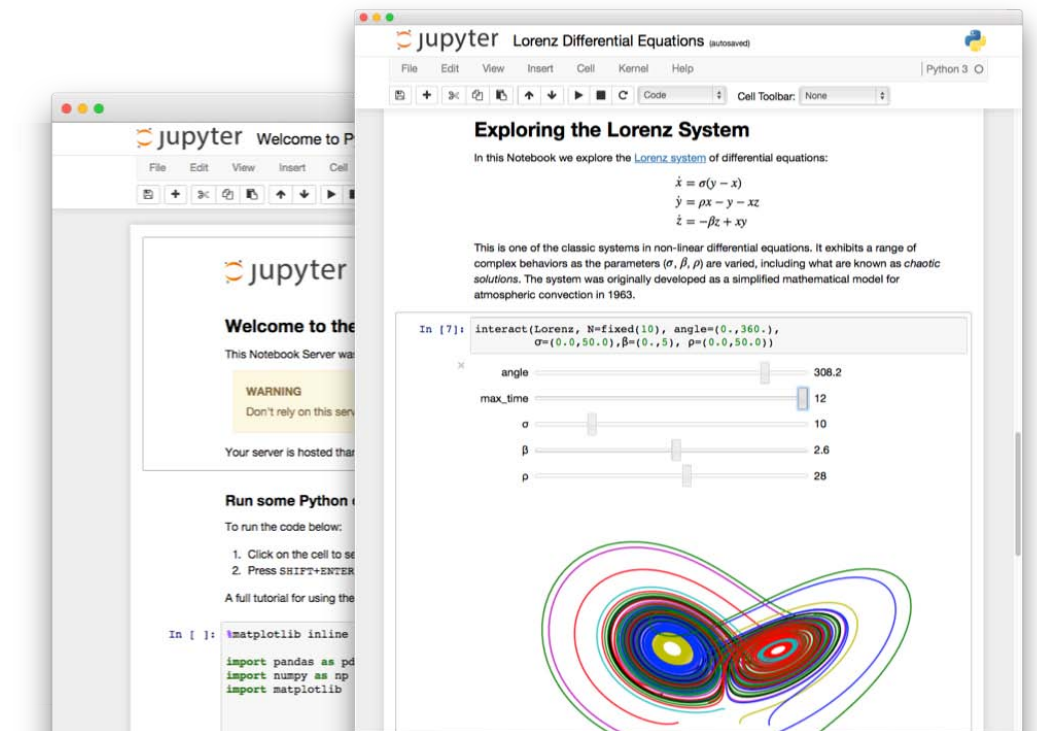


Site

<http://scikit-learn.org/stable/>

3.3 The Jupyter Notebook

The jupyter notebook is an open source web based application that combines the properties of a web page, an office viewer like powerpoint or libreoffice Impress, and an Integrated Development Environment -IDE-. Also it is a very interactive and dynamic environment as concerns the changes on text, images or code. It is used extensively in this thesis, combined with Ipython.



3.4 Graphviz and NetworkX

Both, Graphviz and NetworkX are related to graphs. Graphviz is an open source visualization package for designing graphs using python. The formats that it produces as a final output, are images and SVG for web pages, PDF or Postscript.

According to the official site : “*NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.*”

Logo



Site

Graphviz

<http://www.graphviz.org/>

NetworkX

<https://networkx.github.io/>

3.5 Installation process of Python and Jupyter notebook

The OS which is used is a Debian based system which is widely known as Ubuntu OS. Practically it is a Linux. The version which is used is the 16.04 LTS. The Python which is used is the 2.7 Anaconda Python. The 3.5 is not used by this project because it is not compatible with the Network X. As concerns the Jupyter notebook it is the latest version. Also the version of the NetworkX is the 1.11. The following table shows the tools which are used, and the official download web sites.

3.5.1 Installation of the Python Anaconda 2.7 version

If Python Anaconda 2.7 or 3.5 is installed then the Jupyter notebook is already installed, so the following lines shows how to install Python Anaconda 2.7.

1st step: Python Anaconda 2.7 must be downloaded from the official web site. Please visit:

<https://www.continuum.io/downloads>

2nd step: From the console with the command `cd directory` move to the folder where the downloaded file is.

3rd step: run the command `bash Anaconda2-4.3.1-Linux-x86_64.sh`

```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532:~$ bash Anaconda2-4.3.1-Linux-x86_64.sh
```

Then follow the procedure which is given in the following pictures.

```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532: ~
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532:~$ bash Anaconda2-4.3.1-Linux-x86_64.sh

Welcome to Anaconda2 4.3.1 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Where says do do you wish the installer to prepend the Anaconda2 install location, it is strongly recommended to answer yes, except if you know what you are doing,very well.

```
Python 2.7.13 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Anaconda2 install location
to PATH in your /home/alexandridis_postgraduate/.bashrc ? [yes|no]
[no] >>>
```

```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532: ~
installing: zlib-1.2.8-3 ...
installing: anaconda-4.3.1-np111py27_0 ...
installing: conda-4.3.14-py27_0 ...
installing: conda-env-2.6.0-0 ...
Python 2.7.13 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Anaconda2 install location
to PATH in your /home/alexandridis_postgraduate/.bashrc ? [yes|no]
[no] >>> yes

Prepending PATH=/home/alexandridis_postgraduate/anaconda2/bin to PATH in /home/a
lexandridis_postgraduate/.bashrc
A backup will be made to: /home/alexandridis_postgraduate/.bashrc-anaconda2.bak

For this change to become active, you have to open a new terminal.

Thank you for installing Anaconda2!

Share your notebooks and packages on Anaconda Cloud!
Sign up for free: https://anaconda.org

alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532:~$
```

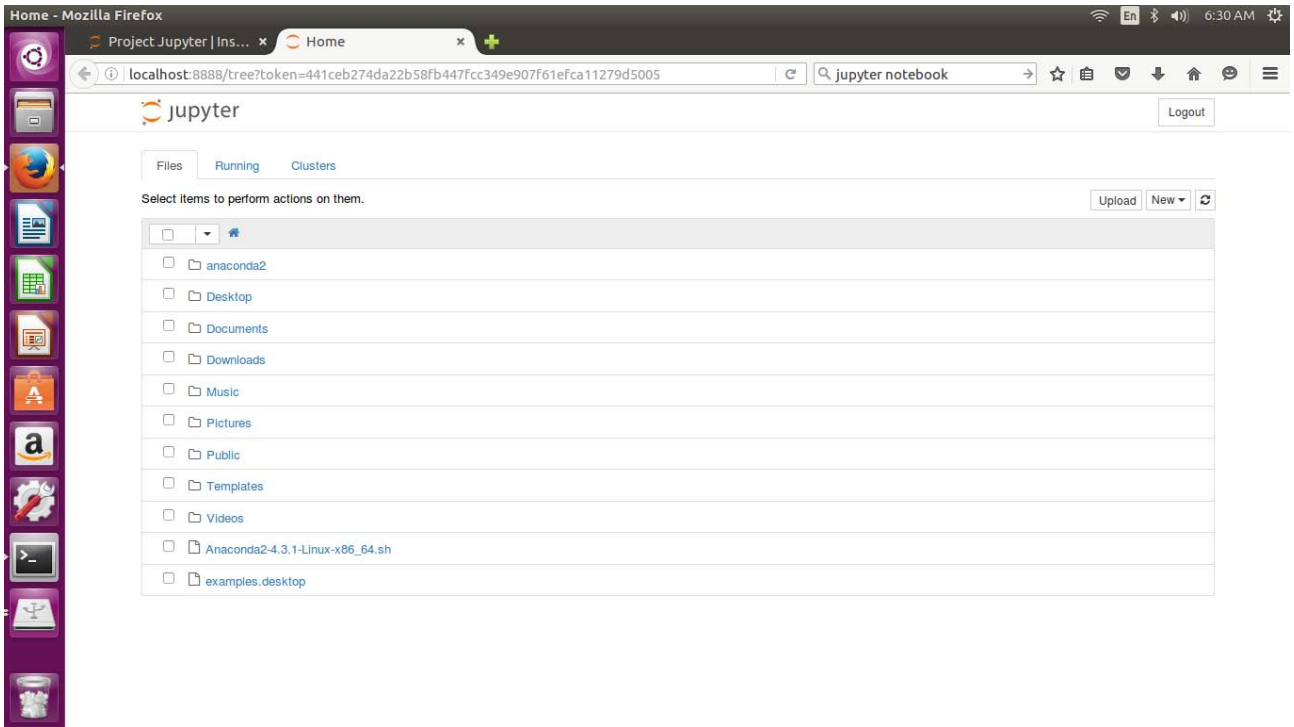
As we can see in the above picture the Anaconda has been installed in our system. Now to run Jupyter Notebook you must follow the order of the following line

3.5.2. Starting the Jupyter Notebook

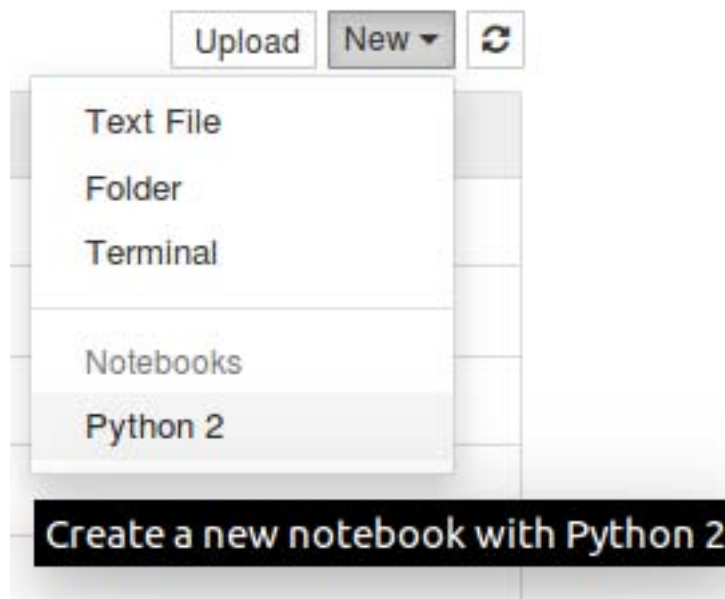
To start the Jupyter Notebook give the console the command *jupyter notebook*.

```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532:~$ jupyter no
tebook
```

It will start the preselected browser automatically and will open the main page of the Jupyter, which gives the directories of the preselected installation file of the Python Anaconda.



To start a new Python Project just follow New → Python. This will start a new Ipython Notebook.



Note: It Must be notice here that the Files of Jupyter are Ipython files. In a way Jupyter is the evolution of Ipython.

To shut Down Jupyter Notebook we open the terminal which has the running commands, and press ctrl+C, and then y and Enter in 5 sec.

```
alexandridis@Postgraduate:~$ jupyter notebook
[I 08:51:41.924 NotebookApp] Writing notebook server cookie secret to /run/user/1000/jupyter/notebook_cookie_secret
[I 08:51:43.002 NotebookApp] Serving notebooks from local directory: /home/alexandridis
[I 08:51:43.002 NotebookApp] 0 active kernels
[I 08:51:43.002 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=f6c0bc3e71fc0c203bc84ae7eae7fbf34aefdcfb7598e27a
[I 08:51:43.003 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 08:51:43.004 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time, to login with a token:
http://localhost:8888/?token=f6c0bc3e71fc0c203bc84ae7eae7fbf34aefdcfb7598e27a
[I 08:51:43.213 NotebookApp] Accepting one-time-token-authenticated connection from 127.0.0.1
^C[I 08:51:56.371 NotebookApp] interrupted
Serving notebooks from local directory: /home/alexandridis
0 active kernels
The Jupyter Notebook is running at: http://localhost:8888/?token=f6c0bc3e71fc0c203bc84ae7eae7fbf34aefdcfb7598e27a
Shutdown this notebook server (y/[n])? █
```

3.5.3 Install Graphviz and NetworkX

Graphviz is not pre-installed in a lot of cases, and also it is Necessary for NetworkX and the visualization of the formed graphs. Finally, it must be installed after Python Anaconda. In order to install Graphviz:

A terminal must be opened and the command `sudo apt-get install graphviz` must be given.

```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEBOOK-AH532:~$ sudo apt-get install graphviz
```

After Graphviz is installed, the same procedure must be repeated for the NetworkX. The command line is: `pip install networkx`

```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEBOOK-AH532: ~  
alexandridis_postgraduate@alexandridis-postgraduate-LIFEBOOK-AH532:~$ pip instal  
l networkx  
Requirement already satisfied: networkx in ./anaconda2/lib/python2.7/site-packag  
es  
Requirement already satisfied: decorator>=3.4.0 in ./anaconda2/lib/python2.7/sit  
e-packages (from networkx)  
alexandridis_postgraduate@alexandridis-postgraduate-LIFEBOOK-AH532:~$ █
```


4. The technical Aspects of the NLP core

This chapter presents the Natural Language Processing -NLP- core and the relative technical aspects of the installation and the usage of this tool. Also, the encapsulation of the NLP to Python is presented by using Jason tool.

4.1 Description of the NLP core to Python using Json

The NLP-core which stands for Natural Language Processing core, is a toolkit which analyzes given text material linguistically. It has been developed by Standford University and it is a very common toolkit for text analysis. More specifically, it provides a simplified description of a sentence's grammatical relationship by determining which parts of the speech are (subject, object, verb,) [31]. The following table provides the basic features of the Standford NLP core [32].

The main features of the Standford NLP core
base forms of words
parts of speech
Names of subjects or/and objects
Normalize dates, times, numeric quantities
mark up the structure of sentences in terms of phrases and word dependencies
which noun phrases refer to the same entities
indicate sentiment
extract particular or open-class relations between entity mentions
get quotes people said

4.2 Encapsulation of the NLP core to Python using Json

The NLP core is Java tool, so it can't be used by Python in the initial form. In order the NLP core to be transformed into a useful tool for Python, the Json tool is used for the encapsulation of The NLP core into the Python.

4.2.1 The Json

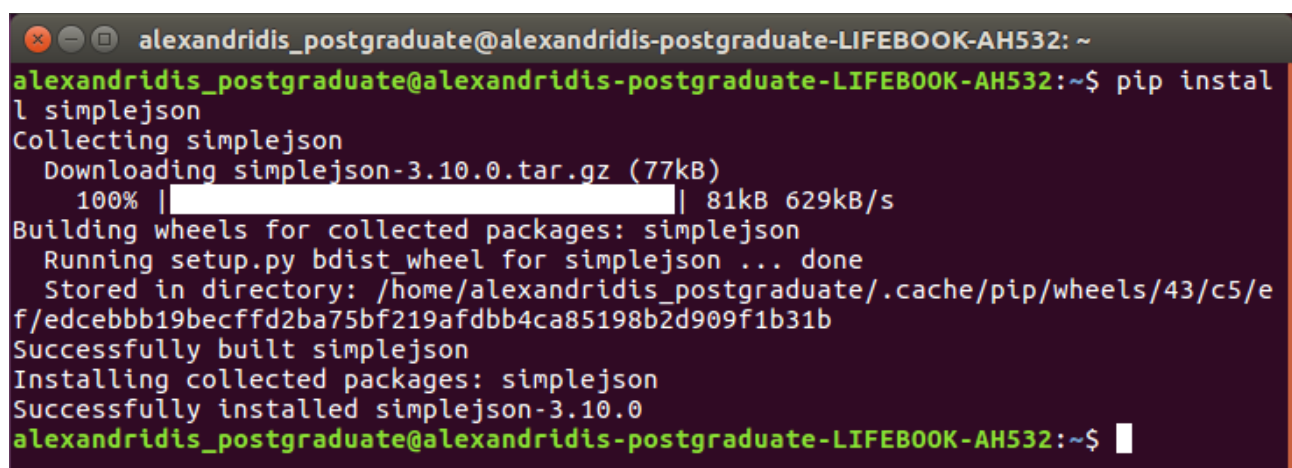
Json which stands for Java Script Object Notation, is a tool for exchanging data in text formation between a browser and a client [30]. It practically transports the data (text) in a Javascript object [30]. At it becomes obvious in this way we can encapsulate NLPCore into Python with little effort.

4.2.2 Installation of NLP Core

First the NLP-core for python version must be downloaded To download it, visit: <https://github.com/dasmith/stanford-corenlp-python>

Then, the json, the jsonrpclib, and the unicode must be installed into the system.

The *pip install* command must be used to install json



```
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532: ~
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532:~$ pip install simplejson
Collecting simplejson
  Downloading simplejson-3.10.0.tar.gz (77kB)
    100% |████████████████████████████████████████| 81kB 629kB/s
Building wheels for collected packages: simplejson
  Running setup.py bdist_wheel for simplejson ... done
  Stored in directory: /home/alexandridis_postgraduate/.cache/pip/wheels/43/c5/ef/edcebbb19becffd2ba75bf219afdbb4ca85198b2d909f1b31b
Successfully built simplejson
Installing collected packages: simplejson
Successfully installed simplejson-3.10.0
alexandridis_postgraduate@alexandridis-postgraduate-LIFEB00K-AH532:~$
```

The same commands are used for the `jsonrpclib` (`pip install jsonrpclib`) and the (`pip install unicode`)

Then, if the above mentioned requirements have been satisfied, we simply run the NLP Core as 4.3.3 mentions.

4.2.3 Commands for encapsulation and running the NLP Core using Json

The following steps present the commands for running NLP core via Python.

1st Step The entrance to the file of the NLP core

```
alexandridis@Postgraduate: ~  
alexandridis@Postgraduate:~$ cd corenlp-python
```

2nd step Run The Script “corenlp.py” in the directory of python corenlp/corenlp.py. The system will run the server and will show the hosting IP.

```
alexandridis@Postgraduate: ~/corenlp-python  
alexandridis@Postgraduate:~$ cd corenlp-python  
alexandridis@Postgraduate:~/corenlp-python$ python corenlp/corenlp.py  
Serving on http://127.0.0.1:8080
```

To exit open the terminal where the NLP core is running and pres Ctrl+C. The system will replay with ^Cbye.

```
alexandridis@Postgraduate: ~/corenlp-python  
alexandridis@Postgraduate:~$ cd corenlp-python  
alexandridis@Postgraduate:~/corenlp-python$ python corenlp/corenlp.py  
Serving on http://127.0.0.1:8080  
^Cbye.  
alexandridis@Postgraduate:~/corenlp-python$
```

For the entrance of Json and NLP Core to our code, we use the commands of the following example, as the server runs on the background. The result is assigned in an object “result”.

```
In [1]: import jsonrpclib
from simplejson import loads
server = jsonrpclib.Server("http://localhost:8080")

result = loads(server.parse("A glowing Kelly Brook showed off her famous midriff as she cheered on new boyfriend
print "Result", result
```

```
Result {'u'coref': [[['u'her', 0, 6, 6, 7], [u'A glowing Kelly Brook', 0, 3, 0, 4]], [['u'she', 0, 10, 10, 11], [u'A glowing Kelly Brook', 0, 3, 0, 4]], [['u'her', 0, 21, 21, 22], [u'A glowing Kelly Brook', 0, 3, 0, 4]], [['u'her', 0, 27, 27, 28], [u'A glowing Kelly Brook', 0, 3, 0, 4]], [['u'her', 0, 44, 44, 45], [u'A glowing Kelly Brook', 0, 3, 0, 4]], [['u'her', 0, 48, 48, 49], [u'her low-key casual look.Only', 0, 47, 44, 48]], [['u'her', 0, 57, 57, 58], [u'her low-key casual look.Only', 0, 47, 44, 48]], [['u'she', 0, 69, 69, 70], [u'her low-key casual look.Only', 0, 47, 44, 48]], [['u'her', 0, 73, 73, 74], [u'her low-key casual look.Only', 0, 47, 44, 48]]], 'u'sentences': [{'u'parsetree': u'(ROOT (S (S (NP (DT A) (JJ glowing) (NNP Kelly) (NNP Brook)) (VP (VBD showed) (PP (IN off) (NP (PRP$ her) (JJ famous) (NN midriff))) (SBAR (IN as) (S (NP (PRP she)) (VP (VP (VBD cheered) (PP (IN on) (NP (JJ new) (NN boyfriend) (NNP Danny) (NNP Cipriani))) (PP (IN from) (NP (NP (DT the) (NNS stands)) (NP (NNP yesterday.With)))) (NP (NP (PRP$ her) (NN brunette) (NNP Hollywood) (NN hair)) (VP (VBG tumbling) (PRT (RP down)) (NP (PRP$ her) (NNS shoulders)))))) (, ,) (CC and) (VP (ADVP (RB carefully)) (VBD applied) (NP (NN make-up)))))) (, ,) (S (NP (NNP Kelly)) (ADVP (RB certainly)) (VP (VBD stood) (PRT (RP out)) (PP (IN from) (NP (DT the) (NN crowd)))))) (, ,) (S (PP (IN despite) (NP (PRP$ her) (JJ low-key) (JJ casual) (NNP look.Only))) (NP (NP (PRP$ her) (JJ ruby) (JJ red) (NNS talons)) (PRN (: -) (CC and) (NP (NP (DT a) (NN flash)) (PP (IN of) (NP (PRP$ her) (JJ famous) (NN tummy)))) (: -))) (VP (VBD gave) (NP (NP (DT a) (NN hint)) (PP (IN of) (NP (DT the) (JJ old) (NN school) (NN glamour)))) (PP (NP (PRP she) (RB usually) (NNS sports)) (IN on) (NP (PRP$ her) (NNS nights))) (ADVP (RP out)))) (. .))', u'text': u'A glowing Kelly Brook showed off her famous midriff as she cheered on new boyfriend Danny Cipriani from the stands yesterday.With her brunette Hollywood hair tumbling down her shoulders, and carefully applied make-up, Kelly certainly stood out from the crowd, despite her low-key casual look.Only her ruby red talons, and a flash of her famous tummy, gave a hint of the old ce
```

More information can be found on the README.md file on <https://github.com/dasmith/stanford-corenlp-python>, and <https://pypi.python.org/pypi/corenlp-python>.

5. The development of the computational method and experiments

This chapter presents the development of the computational method of the Conway's hierarchical model for the Autobiographical memory. All the steps, from the extraction of the Event Specific Knowledge -ESK- from the posts, to the formation of the themes are presented clearly and with their mathematical and technical details. A small but special design example for maximum testing level is used in a presentable and understandable by the reader way.

5.1 The general overview of the computational method

The following scheme is illustrating the steps which are presented and implemented for the development of the computational method based on Conway's model (Figure 2).

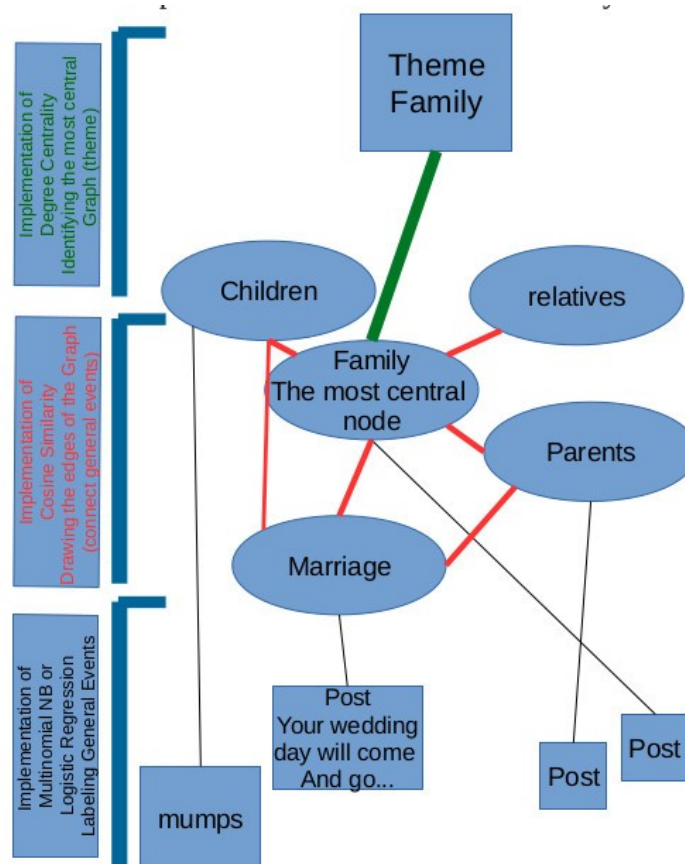


Figure-3, Illustration of the computational method based on Conway's model

5.2 The data sets (Importing data)

The source of the data is my personal account on Facebook, and public data which have derived by using API tools, and manual methods [18-page 255]. Moreover, the anonymity is guaranteed by removing all the data which may reveal the identity of the person. Additionally, some data such as names have been replaced by fictional labels. The final product is the main part of small posts which are relative to the autobiographical memory. The pre-processing of the data is done by hand, and not with an automated method, because, the automated models either will leave fractions of data which are revealing the identity of the person, or will keep a very taugth policy which is damaging the vital information of the data set. The context of [33] has been taken under serious consideration during the pre-processing of the data. Also [18] confirms, in a way, that the best policy as concerns the privacy of the subjects, and the process of the data from the scope of security, is to use manual methods. However, this is possible on the level of a thesis but it is not practically possible to be applied on industrial level, so the provision of an automated pre-processing about the data is required for the data industry.

Such an automated process is not possible to be achieved by algorithms, solely. This is a socio-technical process which demands human interaction. This paper proposes the OpenPDS tools [34] that may be used by the users and the industry to give a solid background as concerns privacy.

5.2.1 Privacy policy

The manual process includes the removal of all the necessary id details by hand. Data which are relevant to the: Name, Surname, any name which indicates location, Nationality, Age and date of birth, other names which are linked with the subject, occupation details, family details and specific patterns of style writing such as mistakes and unique phrases, jargon phrases witch are ether revealing the identity or are contain socking words are removed, or replaced by using fictional labels. The final result has been carefully re-fabricated to keep the syntax and grammar rules in a line that makes sense. As concerns the re-fabrication, it is also a metric of cleaning the data which follows the rules and techniques of the quality of the data [2]. The following table gives specific examples as concerns the strategy and the methods that have been used to produce the final product.

5.2.2 The data files

The data files are Tab Separated Value -TSV- and Comma Separated Value -CSV- files. The tool for the pre-processing about the privacy is the excel (LibreOffice), and the method is visual inspection “by hand”. Also the data set has been reduced to a minimum [18], to make possible the test of specific techniques (Nave Bayes, cosine similarity e.t.c) for specific represented cases as concerns the labeling, and the text classification. However, natural data without limitations has been used to show that the average case has been covered by the proposed combine model, which is used.

5.2.3 Import data

We import a small data set for demonstration purposes, with 2 events, one for marriage and one for friendship. The data set has been formed to test specific cases, as concerns the tokens and the Natural Language processing.

```
In [1]: import pandas as pd
        path = 'data/marriage.tsv'
        dataset = pd.read_table(path, header=None, names=['label', 'message'])
```

The given names for the columns are label and message, respectively.

```
In [2]: dataset.shape
```

```
Out[2]: (57, 2)
```

We deal with 2 columns and 57 rows. So the import posts of our small data is 57 posts relative to marriage and friendship.

```
In [3]: dataset.head(20)
```

```
Out[3]:
```

	label	message
0	marr	marriage
1	marr	married
2	marr	wedding
3	marr	children
4	marr	husband
5	marr	wife
6	marr	family
7	marr	wedded
8	marr	marriage anniversary
9	marr	wedding anniversary
10	marr	Wishing you a lifetime of love and happiness.
11	marr	Your wedding day will come and go, but may you...
12	marr	Best wishes on this wonderful journey, as you ...
13	marr	May the years ahead be filled with lasting joy.
14	marr	May the love you share today grow stronger as ...
15	marr	May your joining together bring you more joy t...
16	marr	May today be the beginning of a long, happy li...
17	marr	Thank you for letting us/me share in this joyf...
18	marr	Wishing you joy, love and happiness on your we...
19	marr	May the love and happiness you feel today shin...

According to the manual (by hand) classification, the class of marriage has 36 posts, and the class of friendship has 21, so the total number is 57.

```
In [4]: dataset.label.value_counts()
```

```
Out[4]: marr      36  
       frien      21  
       Name: label, dtype: int64
```

The numerical values of the class labels are necessary for the process by the Scikit Learn (The special tool for machine learning which is used by python Anaconda).

```
In [6]: dataset.head(10)
```

```
Out[6]:
```

	label	message	label_num
0	marr	marriage	0
1	marr	married	0
2	marr	wedding	0
3	marr	children	0
4	marr	husband	0
5	marr	wife	0
6	marr	family	0
7	marr	wedded	0
8	marr	marriage anniversary	0
9	marr	wedding anniversary	0


```
In [5]: dataset['label_num'] = dataset.label.map({'marr':0, 'frien':1})
```

The main data set is separated into 2 data subsets, according to the message and the label.

```
In [7]: X = dataset.message
        y = dataset.label_num
        print(X.shape)
        print(y.shape)
(57,)
(57,)
```

We split the data into a training subset and a test subset. The training subset will be used to train our models. As has been mentioned before the reason why the label matching has been done by hand [18-page255], is to give more clear and accurate training data to our model. Since we deal with supervised learning a well trained model is vital to achieve higher accuracy and to avoid basic bias errors because of the quality of data [33]. A wrong label could train the model with a wrong prototype especially the Naive Bayes which is a conditional probabilistic model of pairs.

```
In [8]: from sklearn.cross_validation import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
        print(X_train.shape)
        print(X_test.shape)
        print(y_train.shape)
        print(y_test.shape)
(42,)
(15,)
(42,)
(15,)
```

5.3 Vectorization of the data

The vectorization of tokens (words) are their numerical representation for computational reasons.

5.3.1 Unigram Language Model

Python and classification models are not in position to process data which are out of the sphere of the numerical logic. This means that the data must be transformed into numerical entities to be processed. The vectorization of the data is transforming the string formation of the data into numerical values. More specifically the Vectorization method which is applied in our case and is

known as CountVectorizer, is linked the unigram language model, also known as “bag of words”[18-page 117].

According to the bag of words (vectorization) three main techniques take place, “tokenization”, “counting” and “normalization”[20].

In lexical analysis, tokenization is the method which is breaking a text into smaller meaningful forms (e.g. sentences, words), which are known as tokens [18-page 22][20]. In our case the tokens are words.

As concerns the counting or occurrence counting it means that we measure the occurrence of a word into a document.

Last but not least, the normalization process is normalizing and weighing the tokens which are occur in a corpus of documents. The most simplified method is to make the weight be equal to the number of occurrences of a term in a document [18- page 117].

The CountVectorizer transforms a document into a token vector and more specifically into to a sparse array. This is achieved by implementing “tokenization” as well as “occurrence counting” with the same method (CountVectorizer).

```
In [9]: from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
```

```
In [10]: vect.fit(X_train)
X_train_dtm = vect.transform(X_train)
```

```
In [11]: X_train_dtm = vect.fit_transform(X_train)
```

```
In [12]: X_train_dtm
```

```
Out[12]: <42x142 sparse matrix of type '<type 'numpy.int64''>'
with 295 stored elements in Compressed Sparse Row format>
```

```
In [13]: X_test_dtm = vect.transform(X_test)
X_test_dtm
```

```
Out[13]: <15x142 sparse matrix of type '<type 'numpy.int64''>'
with 58 stored elements in Compressed Sparse Row format>
```

The unigram language model presents some limits as concerns the complexity of the language, because it is not in position to analyze a text grammatically, syntactically and finally to represent the semiology of a document, effectively. The order of the words is irrelevant with the context so texts with identical vocabulary and different grammar and syntax will give the same probability. However, it is a very effective technique as concerns our goal which is the structural representation of the autobiographical memory because it is effective in determining that the main

context of two documents is the same or it is represented by a specific predefined label [18-page 241].

5.3.2 Sparsity

The sparsity is limited by the feature selection and the process phase, but for industrial use this is impossible because the corpus will be larger and the unique words could even reach the 1000. The problem is that the repetition of the initial vocabulary exceeds the 10 times (10000 words), because we deal with very common daily vocabulary. The final product will be a very large sparse matrix that consumes large amounts of computational force.

The usual formats which are used in such cases, are: compressed sparse column -CSC- format, Compressed sparse row format -CSR-, matrix coordinate format -COO- [20] [30].

5.4 The multinomial Naive Bayes

In the case of multinomial data such as text classification, the form of the multinomial Naive Bayes is a very common choice.

According to the multinomial version the parametric variable Θ_y is a vector $(\theta_{y1}, \theta_{y2}, \dots, \theta_{yi}, \dots, \theta_{yn})$ for class y , which indicates the conditional probability of each feature i for a specific class y [20].

The main philosophy is the frequency counting, so the parameter is estimated by using the maximum likelihood adopted to the frequency counting [20].

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

By saying frequency counting we mean the repetition of a feature i in the training set T .

$$N_{yi} = \sum_{x \in T} x_i$$

The above term represents the total number of all features in a class.

$$N_y = \sum_{i=1}^{|T|} N_{yi}$$

An example of a pair <document, class> is:

<Your wedding day will come and go, but may your love forever grow, marriage>.

The pair will be checked for common words such as wedding and love, that belong to the class marriage and have been identified during the training phase. The check will use the repetition of each word according to the count frequency as a probabilistic metric of maximum likelihood.

Note: The Laplace and Lidstone factors are working as barriers for any possible zero probability[20].

Also the problem with the maximum Likelihood is that the estimator between 2 terms which are not occurred in the training set is 0 [18- page 259].

<document,class> = <,May today be the beginning of a long, happy life together, marriage>

In the above mentioned example, if the document (post) is not included in the training data set then the estimator will estimate that there is no relationship between the document (post) and the class with label, “marriage”.

```
In [14]: from sklearn.naive_bayes import MultinomialNB
         nb = MultinomialNB()
```

We run the imported nb which stands for Naive Bayes model.

```
In [15]: %time nb.fit(X_train_dtm, y_train)
```

```
CPU times: user 8 ms, sys: 0 ns, total: 8 ms
Wall time: 4.62 ms
```

```
Out[15]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
In [16]: y_pred_class = nb.predict(X_test_dtm)
```

5.5 Confusing error for Naive Bayes

The metric which is used to compute the right and wrong classified documents is the accuracy_scope, which calculates the accuracy of the correct predictions. The returned scale varies from 0 which indicates that there is not even a single correct match of a label and a document, and 1 which indicates that all the predicted documents are matched correctly[20].

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

: \hat{y}_i The predicted value of the i th sample

y_i : The corresponding true value

```
In [17]: from sklearn import metrics
metrics.accuracy_score(y_test, y_pred_class)
```

```
Out[17]: 0.7333333333333328
```

The accuracy model for evaluating the accuracy of the statistic classification model according to the given data.

```
In [18]: metrics.confusion_matrix(y_test, y_pred_class)
```

```
Out[18]: array([[6, 1],
               [3, 5]])
```

```
In [19]: # print message text for the false positives (married 0 incorrectly classified as friend 1)
X_test[y_test < y_pred_class]
```

```
Out[19]: 3    children
Name: message, dtype: object
```

```
In [20]: # print message text for the false negatives (friend 1 incorrectly classified as married 0)
X_test[y_test > y_pred_class]
```

```
Out[20]: 45    homey
         40    amigo
         39    buddies
Name: message, dtype: object
```

```
In [21]: X_test[45]
```

```
Out[21]: 'homey'
```

5.6 Evaluation of Multinomial Naive Bayes using the ROC Curve

According to the Receiver Operating Characteristic -ROC- curve there is a threshold parameter and the document is classified as positive if it belongs to class and negative if it does not.

$$\text{TPR}(T) = \int_T^{\infty} f_1(x) dx$$

TRP- True positive rate

$$\text{FPR}(T) = \int_T^{\infty} f_0(x) dx$$

False Negative Rate -FPR-

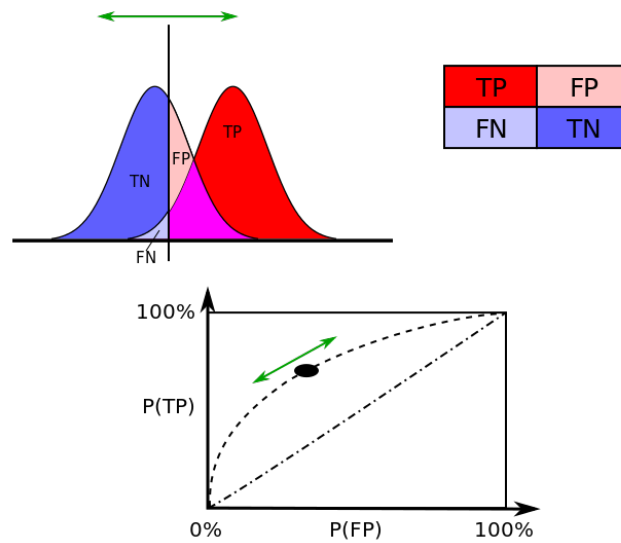


Figure 4- An intuitive representation of the Area under the curve [35]

If the area under the curve is 1 it means that the two distributions are not overlapping and the result is perfect. If the area under the curve is 0.5, then the distributions are overlapping, (practically fit perfectly), which indicates that the model is based on randomness and is completely useless and inaccurate.

To see the function ROC curve in an interactive way, visit: <https://kennis-research.shinyapps.io/ROC-Curves/>

```
In [23]: metrics.roc_auc_score(y_test, y_pred_prob)
```

```
Out[23]: 0.9017857142857143
```

In our case the result is 0.9 which is a very good result if we consider the results of the [13] for the 5 Naive Bayes versions.

5.7 Logistic Regression

First the model of the Max Entropy is used as generalized type of the logistic regression [36]. According to logistic regression, the model calculates the conditional probability:

$$P_w(y|x) \equiv \frac{\exp(w^T f(x, y))}{\sum_{y'} \exp(w^T f(x, y'))},$$

x is the context, y is the label, and w is the weight vector [36].

The logistic regression is a form of the Maximum Entropy which calculates the conditional probability:

$$\mathcal{P}_{\mathbf{w}}(y = \pm 1 | \mathbf{x}) \equiv \frac{1}{1 + e^{-y\mathbf{w}^T \mathbf{x}}},$$

where the \mathbf{x} is the data (context), y is the class label, and \mathbf{w} is the weight vector. Practically the logistic regression is minimizing the logarithmic likelihood between two classes which is represented by:

$$\min_{\mathbf{w}, c} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^n \log(\exp(-y_i(X_i^T \mathbf{w} + c)) + 1).$$

It is known that the logistic regression is very common method in document classification and Natural language processing [36].

In this part as concerns the code, everything is the same with the Multinomial naive Bayes, except that the model now is the logistic regression.

```
In [24]: from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()

We run the logistic regression for the same subset

In [25]: %time logreg.fit(X_train_dtm, y_train)
CPU times: user 4 ms, sys: 4 ms, total: 8 ms
Wall time: 69.2 ms

Out[25]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False)
```

In the following lines the logistic regression is used to predict the values of the classes based on the training set.

```
In [26]: y_pred_class = logreg.predict(X_test_dtm)

In [27]: y_pred_prob = logreg.predict_proba(X_test_dtm)[:, 1]
y_pred_prob
Out[27]: array([ 0.49108172,  0.34474783,  0.0496513 ,  0.46191466,  0.5246418 ,
                 0.11428351,  0.46191466,  0.04590369,  0.0963326 ,  0.56044635,
                 0.67331351,  0.49606193,  0.08541797,  0.46191466,  0.56653985])
```

5.8 Evaluation of Logistic Regression using the ROC curve

Like in the Multinomial Naive Bayes model the `accuracy_scope` and the ROC is used for evaluation of the model.

```
In [28]: metrics.accuracy_score(y_test, y_pred_class)
Out[28]: 0.7333333333333328

In [29]: metrics.roc_auc_score(y_test, y_pred_prob)
Out[29]: 0.9285714285714286
```

As we can see the numbers are almost identical with the Multinomial Naive Bayes (0.73 with the `accuracy_scope` and 0.90 with the ROC curve).

5.9 Frequency Ratio

The frequency ratio is practically based on the repetition of a token and the total number of tokens.

5.9.1 Token counting

We build a corpus of tokens (features) for the calculation of the ratio based on the frequency of each word (this is the training set). This phase belongs to the tokenization process [18-page 22] [20].


```
In [30]: X_train_tokens = vect.get_feature_names()
len(X_train_tokens)
```

```
Out[30]: 142
```

```
In [31]: print(X_train_tokens[0:50])
```

```
['add', 'aged', 'ahead', 'all', 'and', 'anniversary', 'are', 'as', 'be', 'beautifully', 'begin', 'beginning', 'best', 'bless', 'both', 'bring', 'brought', 'buddy', 'build', 'but', 'call', 'can', 'children', 'cliff', 'cobber', 'come', 'congrats', 'congratulations', 'couldn', 'day', 'deepen', 'do', 'drink', 'eat', 'embark', 'enrich', 'even', 'faces', 'family', 'fella', 'filled', 'for', 'forever', 'friend', 'friends', 'fun', 'future', 'george', 'get', 'go']
```

```
In [32]: print(X_train_tokens[-50:])
```

```
['not', 'of', 'off', 'old', 'on', 'one', 'our', 'pal', 'share', 'she', 'signs', 'since', 'stronger', 'survived', 'tell', 'test', 'than', 'thank', 'thanks', 'the', 'this', 'throughout', 'time', 'to', 'today', 'together', 'too', 'union', 'us', 'we', 'wedded', 'wedding', 'welcome', 'were', 'what', 'while', 'who', 'wife', 'will', 'wish', 'wishes', 'wishing', 'with', 'won', 'wonderful', 'would', 'wrinkles', 'years', 'you', 'your']
```

The following part shows the occurrence counting, which means how many times the token is repeated. These are the features.

```
In [33]: nb.feature_count_
```

```
Out[33]: array([[ 1.,  1.,  1.,  1., 13.,  3.,  1.,  4.,  4.,  1.,  1.,
  1.,  4.,  2.,  1.,  1.,  1.,  0.,  1.,  2.,  1.,  1.,
  0.,  0.,  0.,  1.,  2.,  3.,  2.,  3.,  1.,  0.,  1.,
  1.,  1.,  1.,  0.,  1.,  4.,  0.,  2.,  4.,  1.,  1.,
  0.,  1.,  2.,  0.,  1.,  1.,  1.,  3.,  2.,  2.,  4.,
  1.,  1.,  1.,  1.,  1.,  0.,  2.,  1.,  2.,  0.,  1.,
  1.,  1.,  0.,  0.,  1.,  1.,  3.,  1.,  0.,  1.,  1.,
  2.,  1.,  2.,  3.,  6.,  6.,  2.,  0.,  7.,  1.,  1.,
  2.,  1.,  0.,  3.,  1.,  4.,  0.,  1.,  6.,  2.,  1.,
  0.,  2.,  0.,  1.,  0.,  1.,  1.,  0.,  1.,  1.,  1.,
  1.,  9.,  3.,  1.,  1.,  6.,  3.,  8.,  0.,  2.,  2.,
  4.,  1.,  4.,  1.,  0.,  0.,  1.,  1.,  2.,  1.,  1.,
  3.,  2.,  1.,  0.,  3.,  0.,  1.,  2., 16., 14.],
 [ 0.,  0.,  0.,  0.,  1.,  0.,  3.,  0.,  0.,  0.,  0.,
  0.,  1.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,
  1.,  1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,
  0.,  0.,  0.,  1.,  0.,  0.,  1.,  0.,  1.,  0.,  2.,
  4.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  1.,  1.,  0.,
  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,
  0.,  0.,  2.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,
  2.,  0.,  1.,  0.,  1.,  0.,  0.,  1.,  0.,  0.,  0.,
  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
  2.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  0.,
  0.,  0.,  0.,  1.,  0.,  1.,  0.,  0.,  2.,  2.]])
```

```
In [34]: nb.feature_count_.shape
```

```
Out[34]: (2, 142)
```

The following part presents the counting of tokens according to classes.

```
In [35]: marr_token_count = nb.feature_count_[0, :]
marr_token_count
```

```
Out[35]: array([[ 1.,  1.,  1.,  1., 13.,  3.,  1.,  4.,  4.,  1.,  1.,
  1.,  4.,  2.,  1.,  1.,  1.,  0.,  1.,  2.,  1.,  1.,
  0.,  0.,  0.,  1.,  2.,  3.,  2.,  3.,  1.,  0.,  1.,
  1.,  1.,  1.,  0.,  1.,  4.,  0.,  2.,  4.,  1.,  1.,
  0.,  1.,  2.,  0.,  1.,  1.,  1.,  3.,  2.,  2.,  4.,
  1.,  1.,  1.,  1.,  1.,  0.,  2.,  1.,  2.,  0.,  1.,
  1.,  1.,  0.,  0.,  1.,  1.,  3.,  1.,  0.,  1.,  1.,
  2.,  1.,  2.,  3.,  6.,  6.,  2.,  0.,  7.,  1.,  1.,
  2.,  1.,  0.,  3.,  1.,  4.,  0.,  1.,  6.,  2.,  1.,
  0.,  2.,  0.,  1.,  0.,  1.,  1.,  0.,  1.,  1.,  1.,
  1.,  9.,  3.,  1.,  1.,  6.,  3.,  8.,  0.,  2.,  2.,
  4.,  1.,  4.,  1.,  0.,  0.,  1.,  1.,  2.,  1.,  1.,
  3.,  2.,  1.,  0.,  3.,  0.,  1.,  2., 16., 14.]])
```

```
In [36]: frien_token_count = nb.feature_count_[1, :]
frien_token_count
```

```
Out[36]: array([[ 0.,  0.,  0.,  0.,  1.,  0.,  3.,  0.,  0.,  0.,  0.,  1.,
  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.,  1.,  0.,
  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
  1.,  0.,  1.,  0.,  2.,  4.,  0.,  0.,  1.,  0.,  0.,
  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  1.,  1.,
  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  2.,
  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  2.,  0.,  1.,  0.,  1.,
  0.,  0.,  1.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,
  0.,  1.,  0.,  0.,  2.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,
  0.,  0.,  0.,  0.,  0.,  1.,  0.,  1.,  0.,  0.,  2.,  2.]])
```

5.9.2 Token Frequency & Ratio

In this part the frequency and the ratio are calculated according to the counting of tokens. To refresh our memories the counting of the tokens is the number that expresses the repetition of each word. The final products are the frequency and the ratio of the tokens.

```
In [40]: tokens['marr'] = tokens.marr / nb.class_count_[0]
tokens['frien'] = tokens.frien / nb.class_count_[1]
tokens.sample(5, random_state=6)
```

```
Out[40]:
```

	frien	marr
token		
embark	0.0	0.034483
not	0.0	0.034483
journey	0.0	0.034483
deepen	0.0	0.034483
future	0.0	0.068966

The following part calculates the ratio of the tokens, of the friends and marriage classes, respectively.

```
In [41]: tokens['frien_ratio'] = tokens.frien / tokens.marr
tokens.sample(5, random_state=6)
```

```
Out[41]:
```

	frien	marr	frien_ratio
token			
embark	0.0	0.034483	0.0
not	0.0	0.034483	0.0
journey	0.0	0.034483	0.0
deepen	0.0	0.034483	0.0
future	0.0	0.068966	0.0

```
In [42]: tokens.sort_values('frien_ratio', ascending=False)
```

won	0.076923	0.000000	inf
she	0.076923	0.000000	inf
would	0.076923	0.000000	inf
mate	0.076923	0.000000	inf
are	0.230769	0.034483	6.692308
friend	0.153846	0.034483	4.461538
thanks	0.076923	0.034483	2.230769
we	0.153846	0.137931	1.115385
...
filled	0.000000	0.068966	0.000000

I) The “fiends” class

```
In [47]: tokens.sort_values('marr_ratio', ascending=False)
```

inviting	0.000000	0.034483	0.000000	inf
in	0.000000	0.034483	0.000000	inf
imagine	0.000000	0.034483	0.000000	inf
our	0.000000	0.034483	0.000000	inf
...
we	0.153846	0.137931	1.115385	0.896552
thanks	0.076923	0.034483	2.230769	0.448276
friend	0.153846	0.034483	4.461538	0.224138
are	0.230769	0.034483	6.692308	0.149425
tell	0.076923	0.000000	inf	0.000000
mate	0.076923	0.000000	inf	0.000000

II) The “marriage” class

5.9.3. The commonness between the tokens

The following graph presents the commonness of the tokens and the way that it works. The deterministic factor is the ratio.

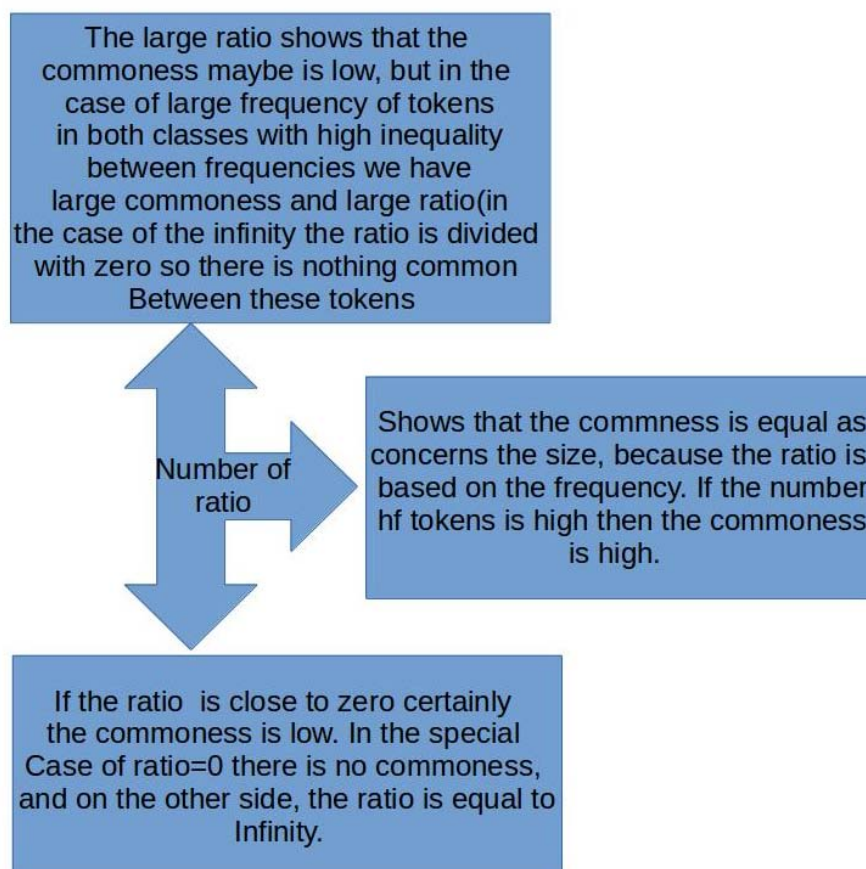


Figure 5- the commonness of the tokens

The following line shows the ratio of the token “marriage” to the class “friends”. It is obvious that the token “marriage” belongs to the “marriage” class, so the ratio 0.0 is a very logical answer.

```
In [43]: tokens.loc['marriage','frien_ratio']  
Out[43]: 0.0
```

The following line shows that the ratio of the token “marriage” for the “marr” class is 0.2, which confirms the above mentioned.

```
In [44]: tokens.loc['marriage','marr']  
Out[44]: 0.20689655172413793
```

5.10 The cosine similarity

The cosine similarity is used to measure the similarity between two documents [21] and more specifically the cohesion between two events, because the similarity is also used to measure the cohesion between clusters [22].

As concerns the main idea, we deal with a metric which measures the cosine of the angle between two non-zero vectors of their dot product [37].

In the case of documents, each token is represented by a vector and the token counting number is a dimensional value of the vector. The document is a multidimensional model which is used to measure the similarity to other documents [22] [37].

Last but not least, the metric of the cosine similarity, measures the orientation and not only the magnitude [22] [37], which makes it an important metric for the similarity between two documents.

The formalization of the Cosine similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

and A_i are B_i components of vector A and B, respectively [38].

For demonstration reasons, we use the manual classification which has the perfect result (100% correct classification). This gives an unbiased error, which provide with us the ability to discover the full potentials of the cosine similarity.

The following line shows the importing of the data and the classes of “marriage” and “friends”.

```
In [1]: import pandas as pd
path = 'data/marriage.tsv'
dataset = pd.read_table(path, header=None, names=['label', 'message'])
```

```
In [2]: dataset.head(10)
```

```
Out[2]:
```

	label	message
0	marr	marriage
1	marr	married
2	marr	wedding
3	marr	children
4	marr	husband
5	marr	wife
6	marr	family
7	marr	wedded
8	marr	marriage anniversary
9	marr	wedding anniversary

```
In [40]: type(dataset)
```

```
Out[40]: pandas.core.frame.DataFrame
```

```
In [3]: dataset.label.value_counts()
```

```
Out[3]: marr    36
frien    21
Name: label, dtype: int64
```

As can be seen by the following lines, the final product is a sparse matrix which contains the cosines of the elements also known as tokens or words.

```
In [8]: from sklearn.metrics.pairwise import cosine_similarity
final_similarity = cosine_similarity(tfidf_matrix[0:35], tfidf_matrix[36:56])
```

```
In [9]: final_similarity.shape
```

```
Out[9]: (35, 20)
```

```
In [10]: print(final_similarity)
```

```

0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. ]
[ 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
0.06126393 0.05133355 0.0435554 0. 0. 0.
0.06006641]
[ 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
0.10628821 0.03330015 0.07556527 0. 0. 0.
0.03896516]
[ 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
0.17568799 0.11290859 0.07070885 0. 0. 0. 0. ]
[ 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0.25005295 0. 0.
0. 0. 0. 0. 0. 0. 0. ]
[ 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0.
0.07629757 0. 0.05424351 0. 0. 0. 0. ]
[ 0. 0. 0. 0. 0. 0. 0.

```

In the following lines the expectation value $E[x]$ of the above sparse metrics is calculated, and represents the Expectation value of the cosine similarity between the two documents.

```
In [13]: temp1 = 0
         for i in range(0,35):
           expectation_1 = final_similarity[i]
           temp = (sum(expectation_1))/20
           print(temp)
           temp1 = temp + temp1
           print("Temp1",temp1)
```

```
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0179442959356
('Temp1', 0.017944295935604065)
0.0
('Temp1', 0.017944295935604065)
0.0222934469676
('Temp1', 0.04023774290316498)
0.0
('Temp1', 0.04023774290316498)
0.0
('Temp1', 0.04023774290316498)
0.0
('Temp1', 0.04023774290316498)
0.0
('Temp1', 0.04023774290316498)
0.0108109641995
('Temp1', 0.05104870710262803)
0.0127059395714
('Temp1', 0.063754646673998133)
0.017965271449
('Temp1', 0.081719918123035723)
0.0125026473441
```

The expectation value for the cosine similarity without further process is 0.0817 while the interval similarity of the “marriage” class is 0.00478 which is a biased answer. The problem is the “single phrase words” which are used in the data set to train, in the best possible way, the Multinomial Naive Bayes. These words in both classes are presenting a cosine similarity of 1 (since they are single word-phrase for training purposes). This creates a biased result as concerns the Expectation values since is an average metric.

In the following lines the cosine similarity is recomputed by only selecting a sample based on the keywords of the data set without these single phrase words for training purposes. The result is a completely different and logical number.

The key words of the dataset

```
In [89]: data_keywords = data.append(keywords, ignore_index=True)
data_keywords
```

```
Out[89]: 0          marriage
1          married
2          wedding
3          children
4          husband
5          wife
6          family
7          wedded
8          marriage anniversary
9          wedding anniversary
10         Wishing you a lifetime of love and happiness.
11         Your wedding day will come and go, but may you...
12         Best wishes on this wonderful journey, as you ...
13         May the years ahead be filled with lasting joy.
14         May the love you share today grow stronger as ...
15         May your joining together bring you more joy t...
16         May today be the beginning of a long, happy li...
17         Thank you for letting us/me share in this joyf...
18         Wishing you joy, love and happiness on your we...
19         May the love and happiness you feel today shin...
20         Congratulations on your wedding!
21         Wishing you a long and happy marriage
22         Here's to a long and happy marriage!
23         Best wishes for a fun-filled future together.
24         Thanks for inviting us to eat and drink while ...
25         May God bless you and your union.
26         May God grant you all of life's blessings and ...
27         May the One who brought you together bless you...
28         Congratulations on your marriage, and welcome ...
29         We/I couldn't be happier to call you both fami...
```

The numerical representation of the key words after the Vectorization.

```
In [90]: tfidf_keywords = tfidf_vectorizer.fit_transform(data_keywords)
print tfidf_keywords.shape
(60, 170)
```

```
In [91]: print tfidf_keywords
```

```
(0, 100)    1.0
(1, 101)    1.0
(2, 151)    1.0
(3, 25)     1.0
(4, 74)     1.0
(5, 157)    1.0
(6, 44)     1.0
(7, 150)    1.0
(8, 100)    0.654542599876
(8, 6)      0.756025121902
(9, 151)    0.687959212708
(9, 6)      0.725749351808
(10, 161)   0.426038584715
(10, 168)   0.260531763023
(10, 94)    0.505324701057
(10, 113)   0.379659179827
(10, 98)    0.333279774939
(10, 5)     0.241941923645
(10, 63)    0.426038584715
(11, 151)   0.246464575412
(11, 98)    0.216356307415
(11, 5)     0.157062219627
(11, 169)   0.389988302617
(11, 34)    0.276572843408
(11, 158)   0.328043267511
:           :
```

The following line presents the final product which is a sparse matrix with the numerical results.

```
In [92]: keywords_similarity = cosine_similarity(tfidf_keywords[57:60], tfidf_keywords[8:57])
```

```
In [93]: keywords_similarity
```

```
Out[93]: array([[ 0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.30558329,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.42681179,  0.         ,
  [ 0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.27621811,
  0.         ,  0.         ,  0.         ,  1.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.33391537,  0.32555792,  0.23707034,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ],
  [ 0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  1.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.         ,
  0.         ,  0.         ,  0.         ,  0.         ,  0.45334257,
  0.         ,  0.28398741,  0.38125691,  0.38579709]])
```

Here the expectation value is calculated for all the cases.

```
In [95]: temp3 = 0
expectation1=0
for i in range(0,3):
    expectation_1 = keywords_similarity[i]
    temp = (sum(expectation_1))/49
    print(temp)
    temp3 = temp + temp3
    print("Temp3",temp3)
```

```
0.0149468383587
('Temp3', 0.014946838358685752)
0.0443420764161
('Temp3', 0.059288914774828907)
0.0511098771447
('Temp3', 0.11039879191956559)
```

```
In [103]: final_keywords_similarity = temp3/49
```

```
In [104]: final_keywords_similarity
```

```
Out[104]: 0.002253036569787053
```

```
In [95]: temp3 = 0
expectation1=0
for i in range(0,3):
    expectation_1 = keywords_similarity[i]
    temp = (sum(expectation_1))/49
    print(temp)
    temp3 = temp + temp3
    print("Temp3",temp3)
```

```
0.0149468383587
('Temp3', 0.014946838358685752)
0.0443420764161
('Temp3', 0.059288914774828907)
0.0511098771447
('Temp3', 0.11039879191956559)
```

```
In [103]: final_keywords_similarity = temp3/49
```

```
In [104]: final_keywords_similarity
```

```
Out[104]: 0.002253036569787053
```

The

final cosine similarity is 0.0022 which is 40 times lower than the initial expectation value. This

answer is very logical and very low indicating that there is very little common things between the two classes.

Moreover the problem will be reduced if the common words, known as “stop words” (and,or, is e.t.c.), are not included in the data set, but even in this form the result is quite satisfactory.

5.11 The centrality and the network X

The centrality determinate the most central node of the formed graph based on similarity. In this part the cosine similarity is repeated with some extra code for reasons of demonstration of the final result that is computed by the proposed method.

5.11.1 The types of centrality

There are different approaches as concerns centrality, and four very popular types are the degree, closeness, Eigenvector, Empirical Illustration of Centrality Measures [23]. In this thesis the degree centrality is used, as the most suitable to the problem of detecting the most important node in a community (general events which indicate a theme).

5.11.2 The degree centrality

According to [13] the definition of the degree centrality is: “the degree of a node is the number of ties it has with members of the other node set”. A more practical definition is given in [24] which says: “The degree of node u is the fraction of nodes connected to it.

$$d_v = \frac{deg(v)}{m}, \text{ for } v \in U,$$
$$d_v = \frac{deg(v)}{n}, \text{ for } v \in V,$$

The formulation of the centrality as given in the [24]

$\text{deg}(u)$ represents the degree of node u [24]. Moreover, a very common tactic is the normalization of the centrality is measured by dividing it by the maximum possible degree or value, which is $n-1$ nodes.

Figure 6 graph illustrates the implementation of the degree centrality in a large graph of general events which are represented as communities. The first event (red dot) indicates the theme of the work while the second (yellow dot) indicates the study themes. The two themes are parallelized, which is obvious in figure-2 of Chapter 2.

```
In [37]: # Plot degree centrality.  
call_degree centrality = nx.degree centrality(G)  
colors =[call_degree centrality[node] for node in G.nodes()]  
pos = graphviz_layout(G, prog='dot')  
nx.draw(G, pos, node_color=colors, node_size=300)
```

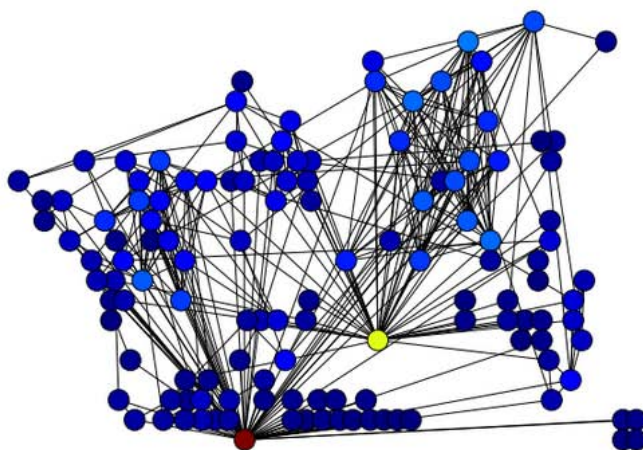


Figure 6- The result of the method in an large graph with a lot of general events

The following graph represents a simplified example which is given for reasons of analytical presentation, as concerns the steps and the methodology which is used to form a graph and to compute the degree centrality. It is worth mentioning that the following graph has been done by hand and it is the desirable result which must be produced by our method, by finding the central family node.

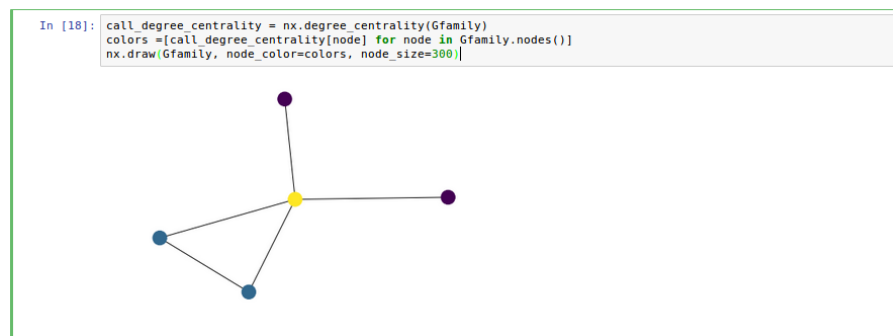


Figure 7 – A prototype result is reproduced correctly and automatically by the following computational methodology which gives the same result (the purple nodes indicate the parent and family general events, the blue nodes the children and marriage, and the yellow node the Family node)

For reasons of analytical demonstration, four different classes are used to form the graph of the family theme. The computation of the cosine similarity is repeated for all the pairs of the classes, to check the possibility of there being an edge between the nodes (classes). The final checks are 10 iterations (5 nodes * 2 ends on each edge equal to n=10).

For brevity's sake we present only the check for two pairs (parents – relatives and family children) and the results for the others, which have an identical code. However, it is vital to repeat the description of the cosine similarity process in a suitable way for the degree centrality, because new information is presented, which is not possible to be presented in a generic description like the previous one.

The following lines import the data set for centrality and assign them in an object dataset_centrality.

```
In [1]: import pandas as pd
path = 'new_data/centrality.tsv'
dataset_centrality = pd.read_table(path, header=None, names=['label', 'message'])

In [2]: type(dataset_centrality)
Out[2]: pandas.core.frame.DataFrame

In [3]: dataset_centrality.label.value_counts()
Out[3]: child    65
famil    24
marr    24
parr    17
relat    8
Name: label, dtype: int64

In [4]: data_centrality = dataset_centrality

In [5]: data_centrality.head()
Out[5]:
```

	label	message
0	marr	marriage
1	marr	married
2	marr	wedding
3	marr	child
4	marr	kid

The data set centrality is not in the right form so it is transformed into a “message” type which is processable by Python.

```

In [6]: data_centrality = dataset_centrality.message
In [7]: data_centrality
Out[7]: 0 marriage
1 married
2 wedding
3 child
4 kid
5 kids
6 children
7 husband
8 wife
9 family
10 wedded
11 marriage anniversary
12 wedding anniversary
13 Wishing you a lifetime of love and happiness.
14 Your wedding day will come and go, but may you...
15 Best wishes on this wonderful journey, as you ...
16 May the years ahead be filled with lasting joy.
17 May the love you share today grow stronger as ...
18 May your joining together bring you more joy t...
19 May today be the beginning of a long, happy li...
20 Thank you for letting us/me share in this joyf...
21 Wishing you joy, love and happiness on your we...
22 May the love and happiness you feel today shin...
23 Congratulations on your wedding!
24 child
25 children
26 twins
27 son
28 sons
29 daughter
...

```

The following line is Vectorizing the data. This means that the data are transformed into numerical values and assigned into an object by the name `tfidf_matrix_centrality`. This object is common for all the cases because it contains all the classes.

```

In [9]: print (tfidf_matrix_centrality)
(0, 103) 1.0
(1, 104) 1.0
(2, 178) 1.0
(3, 35) 1.0
(4, 89) 1.0
(5, 90) 1.0
(6, 37) 1.0
(7, 81) 1.0
(8, 181) 1.0
(9, 57) 1.0
(10, 177) 1.0
(11, 103) 0.707106781187
(11, 6) 0.707106781187
(12, 178) 0.650582241444
(12, 6) 0.759435808425
(13, 185) 0.408924809664
(13, 192) 0.284930028414
(13, 96) 0.44321130385
(13, 118) 0.408924809664
(13, 102) 0.325985005946
(13, 5) 0.365728935521
(13, 74) 0.384598154898
(14, 178) 0.243739926594
(14, 102) 0.226813921233
(14, 5) 0.25446696155
: :

```

The following lines compute the cosine similarity for the “parents-relatives” pair. It is known to us, from the data set inspection, that this pair has not got even a single common word.

This means that the final product must be $\cos = 0$, $\text{angle} = 90^\circ$, $\text{percentage} = 0$. Moreover, as we can see, the final product is a sparse matrix with all its elements equal to 0.

```
In [48]: from sklearn.metrics.pairwise import cosine_similarity
centrality_simil_parr_relat = cosine_similarity(tfidf_matrix_centrality[114:129], tfidf_matrix_centrality[130:136])
```

```
In [49]: centrality_simil_parr_relat.shape
```

```
Out[49]: (15, 8)
```

```
In [50]: centrality_simil_parr_relat
```

```
Out[50]: array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

In this part the system computes the expectation value for all the cases. The final number is 0. This confirms that these two classes have no relationship between them.

```
In [51]: temp1 = 0
for i in range(0,15):
    expectation_parr_relat = centrality_simil_parr_relat[i]
    temp = (sum(expectation_parr_relat))/8
    print(temp)
    temp1 = temp + temp1
    print("Temp1",temp1)
```

```
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
```

Moreover, the percentage is computed and the logical answer is 0.

```
In [52]: import math
cos_test = 0
angle = math.acos(cos_test)
angle_deg = math.degrees(angle)
percentage = (90-angle_deg) *1.11
print ("Percentage of similarity",percentage)
print math.degrees(angle)

('Percentage of similarity', 0.0)
90.0
```

```
In [15]: from sklearn.metrics.pairwise import cosine_similarity
centrality_simil_famil_child = cosine_similarity(tfidf_matrix_centralty[89:112], tfidf_matrix_centralty[24:88])
```

```
In [16]: centrality_simil_famil_child.shape
```

```
Out[16]: (23, 64)
```

```
In [17]: print (centrality_simil_famil_child)
```

```
[[ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]]
```

The following part repeats the process for the “family-children” pair. The steps are identical with the “parents-relatives” pair.

As it can be seen, the final result is $\cos = 0,180$, $\text{degrees} = 79.6^\circ$, and the percentage is 11,51%.

```
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0
('Temp1', 0.0)
0.0321810635884
('Temp1', 0.032181063588351759)
0.015625
('Temp1', 0.047806063588351759)
0.015625
('Temp1', 0.063431063588351766)
0.0
('Temp1', 0.063431063588351766)
0.0233075434822
('Temp1', 0.086738607070574)
0.015625
('Temp1', 0.102363607070574)
0.0
('Temp1', 0.102363607070574)
0.0
('Temp1', 0.102363607070574)
0.0
('Temp1', 0.102363607070574)
0.015625
('Temp1', 0.117988607070574)
0.0
('Temp1', 0.117988607070574)
0.0
('Temp1', 0.117988607070574)
0.015625
('Temp1', 0.13361360707057401)
0.015625
('Temp1', 0.14923860707057401)
0.0
('Temp1', 0.14923860707057401)
0.015625
('Temp1', 0.16486360707057401)
0.015625
('Temp1', 0.18048860707057401)
0.0
('Temp1', 0.18048860707057401)
```

It is worth saying that for the calculation of the percentage the supplementary angle is used because the perfect match is $\cos = 1$ which is 0 degrees.

```
In [19]: import math
cos_test = 0.180
angle = math.acos(cos_test)
angle_deg = math.degrees(angle)
percentage = (90-angle_deg) *1.11
print ("Percentage of similarity",percentage)
print math.degrees(angle)

('Percentage of similarity', 11.510433384079931)
79.6302401945
```

The following table presents the results for all the pairs which are possible edges for drawing.

Edge	Percentage	Degree	cos
Family-parents	30.54	62	0.462
Family-children	11.51	79.6	0.180
Family- relatives	11	80	0.173
Family-marriage	19.37	72.5	0.30
Marriage-children	14.75	76.7	0.23
Marriage-parents	12.35	78.87	0.193
Marriage-relatives	0	90	0
Parents-children	2.29	87.9	0.036
Parents-relatives	0	0	0
Relatives-children	0	90	0

In this face the system computes the average of all the expectation values as a percentage which is a simple but very balanced metric. Anything lower than average is not included in the drawing edges. The code is presented in a simplified form (not a loop) for the clear illustration of the numbers and the nodes.

```
In [58]: #The average of percentage is calculated (the list contains the percentages of the edges)
average_List = [30.54,11.51,11,19.37,14.7,12.35,0,2.29,0,0]
```

```
In [59]: average = sum(average_List)/10
```

```
In [60]: average
```

```
Out[60]: 10.176
```

```
In [152]: #The average of percentage is calculated (the list contains the percentages of the edges)
average_List = [10.28,17.38,2.29,0,13.45,13.7,35.75,10.86,0,0]
```

```
In [155]: average = sum(average_List)/10
```

```
In [156]: average
```

```
Out[156]: 10.370999999999999
```

Finally, the edges which have been drawn are included in the graph, and the centrality is computed. It is obvious that the result is quite satisfactory. The method has identified the central node which is the “family” class. As becomes obvious from Conway’s model (Figure 2) this class represents the Theme “family” in the life-story of this person. This is the scope of this thesis and as it can be seen, the goal has been achieved, even with limited data sets.

```
In [169]: import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
import graphviz as pgv
import random
from networkx.drawing.nx_agraph import graphviz_layout
from IPython.display import Image, display
```

```
In [170]: # The nodes
Gfamily=nx.Graph()
Gfamily.add_node('family')
Gfamily.add_node('parents')
Gfamily.add_node('children')
Gfamily.add_node('marriage')
Gfamily.add_node('relatives')
```

```
In [171]: #The edges included edges
Gfamily.add_edge('family','children')
Gfamily.add_edge('family','relatives')
Gfamily.add_edge('family','marriage')
Gfamily.add_edge('family','parents')
Gfamily.add_edge('marriage','children')
Gfamily.add_edge('marriage','parents')
nx.draw(Gfamily)
```

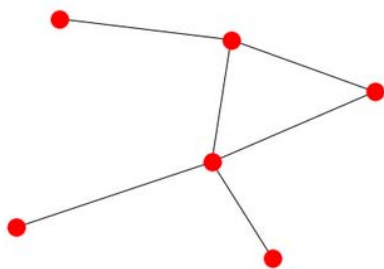


Figure 8 – The produced graph based on the cosine similarity


```
In [64]: call_degree centrality = nx.degree centrality(Gfamily)
colors =[call_degree centrality[node] for node in Gfamily.nodes()]
nx.draw(Gfamily, node_color=colors, node_size=300)
```

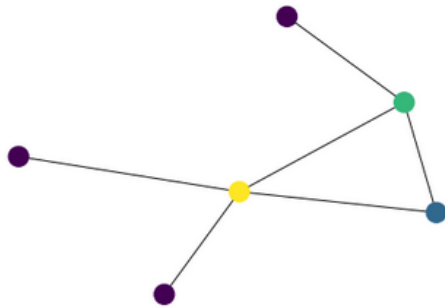


Figure 9- The centrality degree

The yellow dot indicates the central node which is the general event “family” and represents the theme “family”. Similarly to the prototype of Figure 7 which has been done by human, the method has achieved to identify the central node automatically and effectively.

6. Conclusion and future plans

This chapter presents the achievement of our goals, efforts and the future plans for the evolution of this method.

6.1 Conclusion

In general this thesis has achieved its goal by forming a method which determines the user's social and psychological profile. This is possible by developing a computational method of Conway's model for the Autobiographical Memory.

The method has three stages. The first stage is the classification of the raw text material by using two very common methods, the Naive Bayes and the Logistic Regression, which is a specialized form of max Entropy. During the second stage the graphs are formed by using the cosine similarity to check the cohesion between the events which are the nodes of the graph. Finally in the third stage the themes which are represented by the label of the most central node, are determined with the usage of the degree centrality.

Having weighted up all the evidence, it is proved that all stages are performing well in simplified and more complex datasets. It is worth mentioning that the simplified data sets are case selective sets for high performance testing as concerns the accuracy of all the used methodology and techniques. Moreover, the performance has been tested using not only traditionally intuitive techniques such as limited datasets, but widely accepted methods of testing such as the Receiver Operating Characteristic -ROC- curve. Both have had almost a 90% and 70% performance, respectively, which are very high performances as concerns the limited data sets, which have been used. This high performance is confirmed by international bibliography and research papers.

Moreover, this thesis has shown the necessity for future work as concerns the statistical models for syntax and grammatical analysis which are using tools such as the NLP-core.

To conclude, this method provides a powerful solution as concerns the structural formation and the retrieval of data in harmonized way with the real world which is represented by psychological theories such as Conway's model. This is a very useful in a variety of applications such as the sentiment detection because it is the first step for more analytical models. Also, this

method is possible to be used as an input for social analysis algorithms, because it can reduce the bias error due to the structural formation which follows a well established model.

6.2 Future plans

In the future plans this thesis will be designed and developed, as concerns the prior process of the data combined with the automated syntax and grammar analysis of the documents, at a level that will meet the demands of the market. Moreover this further development is also aiming to satisfy in the best possible way, two represented factors of the data, privacy and quality.

During the pre-processing of the data, automated methods which are based on the supervised and unsupervised learning will be developed and fully tested, for the accuracy and the analytical abilities of the semiology of the documents which is vital for a more holistic view of the contextual meaning.

As concerns the privacy, the semiotic dimension of the subjects and objects will be identified and will be automatically replaced by fictional labels which are aiming to cover the identity of a subject. Moreover training algorithms must identify patterns that could reveal the identity such as unique jargon words that the person repeats. Even with the above mentioned automated techniques, we could not say that the identity of a person is safe, so interactive applications such as Open PDS are necessary for better security measures in a personalized framework.

The second main factor is the quality of the data which is determined not only by the type of the words but also by the syntax and grammar analysis of the documents. Tools such as the NLP core will be combined with unsupervised learning algorithms such as clustering algorithms (e.g. k-means) to recognize the contextual meaning of the documents.

All in all, the future planing is focusing on more automated models which are based on state of the art analytical techniques for contextual meaning. This can not be achieved by supervised or unsupervised learning solely, but wherever it is necessary re-enforcement and deep learning will be implemented.

7. Bibliography and References

- [1] Thomas S. Huang Anton Nijholt, Maja Pantic Alex Pentland (Eds.), *Artificial Intelligence for human computing, ICMI 2006 and IJCAI 2007 International Workshops*, LNAI vol. 4451, 2007.
- [2] Li Zhang, Marco Gillies, John A. Barnden, Robert J. Hendley, Mark G. Lee, Alan M. Wallington, *Affect Detection and an Automated Improvisational AI Actor in E-Drama, Artificial Intelligence for Human Computing, LNCS*, vol. 4451, pp. 339-358, 2007.
- [3] Mao, Qiushi., *Experimental Studies of Human Behavior in Social Computing Systems*, Doctoral dissertation, Harvard University, Graduate School of Arts & Sciences, 2015.
- [4] Ron Sun, *The Cambridge Handbook of Computational Psychology*, Cambridge University Press, 2008.
- [5] John Turner, *Episodic, Procedural and Semantic Memory*, tutor 2u
- [6] Larry R. Squire and Stuart Zola, *Episodic memory, Semantic Memory, and Amnesia, Hippocampus*, Vol. 8 (3), pp. 205–211, 1998.
- [7] Martin A. Conway, *Memory and the self*, *Journal of Memory and Language*, Vol. 53 (4), pp. 594–628, 2005.
- [8] Xinzhi Wang, Xiang feng Luo, Jinjun chen, *Social Sentiment Detection of Event via a Microblog*, *IEEE 16th International Conference on Computational Science and Engineering*, pp. 1051-1058, 2013.
- [9] Bo Pang and Lillian Lee, Shivakumar Vaithyanathan, *Thumps up? Sentiment classification using Machine learning techniques*, *EMNLP '02 Proceedings of the ACL-02 conference on Empirical methods in natural language processing – vol.10*, pp. 79-86, 2002.
- [10] Andrew Kachites McCallum, Karnal Nigam, *Employing EM and Pool-Based Active Learning for Text Classification*, *ICML '98 Proceedings of the Fifteenth International Conference on Machine*, pp. 350-358, 1998.
- [11] Andrew Kachites McCallum, Karnal Nigam, *Using Maximum Entropy for Text Classification*, *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pp. 61–67, 1999.

[12] Andrew Kachites McCallum, Karnal Nigam, A Comparison of Event Models for Naive Bayes Text Classification, *1998 AAAI Workshop*, pp. 41–48, 1998.

[13] Vangelis Metsis, Ion Androutsopoulos, Georgios Paliouras, Spam filtering with Naive Bayes – Which Naive Bayes ?, *CEAS 2006 - Third Conference on Email and Anti-Spam*, 2006.

[14] Andrew Kope, Caroline Rose, Michael Katchabaw, Modeling Autobiographical Memory for Believable Agents, *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 23-29, 2013.

[15] Martin A. Conway, Christopher W. Pleydell-Pearce, The construction of Autobiographical Memories in the Self-Memory System, *Psychological review*, Vol. 107 (2), American psychological Association, pp, 262-288, 2000.

[16] Ah-Hwee Tan, Gail A. Carpenter, Stephen Grossberg, Intelligence Through Interaction: Towards a Unified Theory for Learning, *ISNN 2007, Part I, LNCS 4491*, pp. 1094-1103, 2007.

[17] M. A. Conway and H. L. Williams, Autobiographical Memory, In John H. Byrne et al., *Learning and Memory: A Comprehensive Reference*. Oxford: Elsevier Ltd. pp. 893-909, 2008.

[18] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[19] Harry Zhang, The optimality of Naive Bayes, *FLAIRS Conference*, AAAI Press, 2004.

[20] Scikit-learn Documentation www.scikit-learn.org powered by Google.

[21] Singhal, Amit, *Modern Information Retrieval: A Brief Overview*, *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* vol. 24 (4), pp. 35–43, 2001.

[22] P.-N. Tan, M. Steinbach & V. Kumar, *Introduction to Data Mining*, Addison-Wesley, chapter 8, 2005.

[23] Stephen and Danial S. Halgin, *The SAGE Handbook of Social Network Analysis-Chapter 8 Analyzing Affiliation Networks*, LINKS Center for Social Network Analysis, Gatton College of Business and Economics University of Kentucky Lexington, SAGE Publications, 2011.

[24] Aric Hagberg, Dan Schult, Pieter Swart, *Network X, documentation*, <https://networkx.github.io/>, 2013.

[25] Bharath Sriram, David Fuhry, Engin Demir, Hakan Ferhatosmanoglu, Murat Demirbas, Short Text Classification in Twitter to Improve Information Filtering, SIGIR'10, pp. 841-842, 2010.

[26] Guido van Rossum, <http://python-history.blogspot.gr/2009/01/brief-timeline-of-python.html>

[27] Python, wikipedia, [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

[28] Python software foundation, <https://www.python.org/about/>

[29] Jason official site, <http://www.json.org/>

[30] Scipy official site, <https://www.scipy.org/>

[31] Stanford type dependencies manual, Marie-Catherine de Marneffe and Christopher D. Manning, Stanford university.

[32] Stanford NLP Core official site, <https://nlp.stanford.edu/software/>

[33] Mats Bergdahl, Manfred Ehling, Eva Elvers, Erika Földesi, Thomas Körner, Andrea Kron, Peter Lohauß, Kornelia Mag, Vera Morais, Anja Nimmergut, Hans Viggo Sæbø, Ulrike Timm, Maria João Zilhão, Manfred Ehling and Thomas Körner (eds), Handbook of data quality, assessment methods and tools, Eurostat, 2007.

[34] Yves-Alexandre de Montjoye, Erez Shmueli, Samuel S. Wang, Alex Sandy Pentland, openPDS: Protecting the Privacy of Metadata through SafeAnswers, PLoS ONE 9(7): e98790, 2014.

[35] Receiver operating characteristic, wikipedia, https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[36] Hsiang-Fu Yu, Fang-Lan Huang, Chih-Jen Lin, Dual Coordinate Descent Methods for Logistic Regression and Maximum Entropy Models, Machine Learning Journal, vol. 85 (1), pp. 41-75, 2001.

[37] Christian S. Perone, Machine Learning:: Cosine Similarity for Vector Space Models (Part III), <http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/>, 2013.

[38] Cosine similarity, wikipedia, https://en.wikipedia.org/wiki/Cosine_similarity