Integration of Heterogeneous Data Types Using Self Organizing Maps

by

Farid Bourennani

A thesis
presented to the University of Ontario Institute of Technology
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Oshawa, Ontario, Canada, July 2009

Abstract

With the growth of computer networks and the advancement of hardware technologies, unprecedented access to data volumes become accessible in a distributed fashion forming heterogeneous data sources. Understanding and combining these data into data warehouses, or merging remote public data into existing databases can significantly enrich the information provided by these data. This problem is called *data integration*: combining data residing at different sources, and providing the user with a unified view of these data. There are two issues with making use of remote data sources: (1) discovery of relevant data sources, and (2) performing the proper joins between the local data source and the relevant remote databases. Both can be solved if one can effectively identify semantically-related attributes between the local data sources and the available remote data sources. However, performing these tasks manually is time-consuming because of the large data sizes and the unavailability of schema documentation; therefore, an automated tool would be definitely more suitable.

Automatically detecting similar entities based on the content is challenging due to three factors. First, because the amount of records is voluminous, it is difficult to perceive or discover information structures or relationships. Second, the schemas of the databases are unfamiliar; therefore, detecting relevant data is difficult. Third, the database entity types are heterogeneous and there is no existing solution for extracting a richer classification result from the processing of two different data types, or at least from textual and numerical data.

We propose to utilize self-organizing maps (SOM) to aid the visual exploration of the large data volumes. The unsupervised classification property of SOM facilitates the integration of completely unfamiliar relational database tables and attributes based on the contents. In order to accommodate heterogeneous data types found in relational databases, we extended the term frequency – inverse document frequency (TF-IDF) measure to handle numerical and textual attribute types by unified vectorization processing. The resulting map allows the user to browse the heterogeneously typed database attributes and discover clusters of documents (attributes) having similar content.

The discovered clusters can significantly aid in manual or automated constructions of data integrity constraints in data cleaning or schema mappings for data integration.

Acknowledgements

I would like to express my appreciation to my advisory committee: Dr. Ying Zhu, and Dr. Ken Pu. Thanks for giving me the opportunity to do research with you. Thanks Dr. Zhu for having trusted me. Special thanks to Dr. Pu for his time, patience, and extremely valuable scientific advices.

This work would not have been possible without the constant support of my family. Special thanks to my dear mother, and kind wife for having supported me through this journey.

List of Abbreviations

BMU	Best matching unit (in SOM training)
BF	Bin frequency
DM	Data mining
IDBF	Inverse document bin frequency
IDF	Inverse document frequency
IR	Information Retrieval
KDD	Knowledge discovery in databases
ML	Machine learning
PCA	Principal component analysis
RP	Random projection
SOM	Self organizing map
TF	Term frequency
UV	Unified Vectorization
VDM	Visual data mining
VSM	Vector space model

Table of Contents

Abstract ii
Acknowledgements iv
List of Abbreviations v
Table of Contents vi
List of Figures viii
List of Tables ix
Chapter 1 Background and Motivation 10
1.1 Problem overview 10
1.1.1 Contribution of the thesis 11
1.1.2 Outline of the thesis 12
1.1.3 List of publications and the author's contributions 12
Chapter 2 Data Integration 13
2.1 Problem overview 13
2.2 Heterogeneous data processing for data integration 13
2.2.1 Semantic integration problem (semantic conflicts) 14
2.2.2 Ontology based data integration 14
2.2.3 Semantic based data integration 15
2.2.4 Benefits of the SOM for solving data integration heterogeneities 16
Chapter 3 Pre-Processing 20
3.1 Data cleaning 21
3.1.1 Data representation 21
3.1.2 Text tokenization 23
3.1.3 Data transformation 32
3.1.4 Dimensionality reduction 34
3.1.5 Pre-processing review 35
Chapter 4 Processing 37
4.1 Self organizing map 37
4.2 SOM training 38
4.2.1 Training initialization 38
4.2.2 Training modes 38
4.3 Visualization 43
4.3.1 Mapping 43

4.3.2 Advantage of map display 44
4.4 Processing summary 49
Chapter 5 Post-Processing 50
5.1 Common item-set based classifier 50
5.2 Algorithm description 51
5.2.1 Phase 1: Forming item-sets 51
5.2.2 Phase 2: SOM's clusters homogeneity validation 53
5.2.3 Phase 3: Clustering unique documents 55
5.2.4 Phase 4: Remapping the unique documents 56
5.3 SOM's map update 57
5.4 Post-processing overview 58
Chapter 6 Case Study 59
6.1 Corpus 59
6.2 Pre-processing setup 60
6.3 Evaluation measures 61
6.4 Case study one: Northwind 62
6.4.1 Experimental objectives 62
6.4.2 Preliminary tests: without dimension reduction 62
6.4.3 Test series one: with dimension reduction 64
6.5 Use case two: Sakila 66
6.5.1 Experimental objectives 66
6.5.2 Experiments 67
6.6 Summary of experiments 70
Chapter 7 Conclusion 72
Bibliography 73
Image References 77

List of Figures

Figure 1 The used data pre-processing techniques 20
Figure 2 Proposed pre-processing steps 22
Figure 3 Pre-processing steps 35
Figure 4 SOM: Projection of high demensional data into two dimensional space 37
Figure 5 Updating the best matching unit (BMU) 40
Figure 6 Hexagonal neighborhood grid 40
Figure 7 Rectangular neighborhood grid 40
Figure 8 Batch version of SOM 42
Figure 9 Trained SOM map 44
Figure 10 Example of failure to scale for large schema 45
Figure 11 SOM map overview 47
Figure 12 SOM map zooming 47
Figure 13 Coloration of the neighboring clusters. 48
Figure 14 Unbalanced trained SOM map with dissimilar measures 50
Figure 15 SOM's clusters homogeneity validation and redistribution 54
Figure 16 SOM's trained map 57
Figure 17 Updated SOM's map after CIBC post-processing 58
Figure 18 Precision measures with different representations 68
Figure 19 Recall with different representations 69
Figure 20 F-Measure with different representations 69

List of Tables

Table 1 VSM matrix for textual data 23
Table 2 Examples of TF-IDF calculation..... 27
Table 3 Numerical portion of the VSM matrix..... 29
Table 4 Unified vectorization of textual and numerical data..... 33
Table 5 Example term-document matrix..... 52
Table 6 databases used for case studies 60
Table 7 F-measure (preliminary tests) 63
Table 8 Entropy measure (preliminary tests)..... 64
Table 9 F-Score values with dimension reduction..... 65
Table 10 Entropy values with dimension reduction..... 65
Table 11 Precision measure with different representations 67
Table 12 Recall measure with different representations..... 67
Table 13 F-measure with different representations..... 68

Chapter 1
Background and Motivation

1.1 Problem overview

There is much interest from the industry in heterogeneous data classification. This interest seems to be proportional to the heterogeneity of the data, i.e., the more heterogeneous the data types are, and the greater the interest there is in automatically processing it. This is likely due to the availability and abundance of such data. For example, business intelligence sectors and financial institutions are extremely eager to extract coincident clustering from textual and numerical (financial specially) data based on the content. Consequently, several business - and finance related - research projects worked on mining quantitative (numerical) and qualitative (textual) data for comparing companies [1], [2], [3]. Numerical data included the company financial information, and the textual information was business reports from external financial analysts. Some projects try to use heterogeneous data mining for providing tools at the level of strategic decision-making [4], while others use the same logic for automated project management valuation and automated problem detection [5]. There have been studies in using data mining for heterogeneous data classification in medical and biomedical fields to classify for example textual and genetic data [6]. In brief, heterogeneous data mining is of great importance in many industrial fields. However, there is no existing literature reporting successful coincident classification based on the content from heterogeneous data types, likely because of the difficulty of combining results.

In this thesis, in order to do data mining of the heterogeneous numerical and textual data, we propose to use unified vectorization for the data. Unified vectorization permits to simultaneously process these heterogeneous data types for better clustering results, rather than processing them separately, and then trying to combine the mining results, which is extremely complicated. The proposed unified vectorization is a more natural way of mining heterogeneous data types because of the simultaneous processing of the data types which results in a unified view of the data. That is, it is possible to have a unified

semantic view of the different entities despite their heterogeneous data types. It is possible to classify purely textual documents, purely numerical documents, and a mixture of numerical and textual documents, based on their content in a single unified map; thus, it greatly simplifies the data exploration. These explorative qualities simplify user browsing and discovery of the data, particularly when the data is voluminous.

In order to be able to extract coincident clusters from heterogeneous data types, much effort has been dedicated to the pre-processing phase, which is may be even the most important part of the whole process [7]. The Term Frequency – Inverse Document Frequency (TF-IDF) measure was extended in order to handle numerical and textual attribute types by unified vectorization processing using self organizing map (SOM). SOM is an unsupervised neural network method which offers the advantage of producing a mapping of high dimensional input space onto a low-dimensional (usually 2D) map, where similar input data can be found on nearby regions of the map. In essence, we selected SOM because its semantic map topology facilitates the exploration and discovery of content based similarities among different documents.

1.1.1 Contribution of the thesis

- We demonstrate that is possible using SOM to meet the challenge of extracting *richer and convergent* results from h*eterogeneous textual* and *numerical* data types, by extending the traditional text analysis from information retrieval and using statistical methods.
- We propose a new measure named Bin Frequency - Inverse Document Bin Frequency (BF-IDBF) for *numerical data type* that permits the heterogeneous process of textual and numerical data types efficiently for better clusters results.
- We demonstrate that a similar data representation for heterogeneous data mining, like the combination of BF-IDBF and TF-IDF measures, can significantly enhance the classification algorithm performance for heterogeneous data mining by unified vectorization.

- We propose the post-processing algorithm Common Item-set Based Classier (CIBC), that enhances the results obtained from SOM's trained map. More particularly, It enhances the recall (heterogeneity) of clusters which helps differentiating visually, on the SOM trained map, between heterogeneous and homogenous data clusters.

1.1.2 Outline of the thesis

The organization of the remainder of the thesis is as follows. Chapter 2Chapter 1 describes how the proposed techniques fit within data integration scope. Chapter 3 discusses the pre-processing techniques such as unified vectorization. Chapter 4 deals with the processing phase: it describes the two versions of SOM and the visualization features of the proposed integration tool. Chapter 5 is devoted to post-processing algorithm called: common item-set based classifier (CIBC). CIBC serves to improve the precision of the clustering; hence, the quality of the clusters. Finally, the case study and the conclusion are in the Chapter 6 and Chapter 7, respectively.

1.1.3 List of publications and the author's contributions

1. **Bourennani, F.**, Pu, K. Q., Zhu., Y. (2009) Visualization and Integration of Databases using Self Organizing Maps, Proceedings of the International Conference on Advances in Databases, Knowledge, and Data Applications, (DBKDA’09), Cancun, Mexico, pp. 155-160.
2. **Bourennani, F.**, Pu, K. Q., Zhu, Y. (2009) Visual Integration Tool for Heterogeneous Data Type by Unified Vectorization, Proceedings of the 10th IEEE International Conference in Reuse and Integration (IRI’09), Las-Vegas, USA, pp. 132-137.

Chapter 2
Data Integration

2.1 Problem overview

The definition of data integration is the process of combining data residing at different sources and providing the user with a unified view of these data [8]. It emerges when two companies need to merge their databases, or a single company needs to enrich its repository by adding some data from another database (e.g., a public database or another company's repository).

Different approaches have been proposed for data integration [9], [10], [11], [12], [13]. They may be divided into two main categories: *ontology based* versus *semantic based* data integration. The first one emphasizes the database structural model while the second one focuses more on the semantic content of database entities.

2.2 Heterogeneous data processing for data integration

The proposed heterogeneous data mining methods could be used to classify any textual, numerical, or hybrid documents. However, in our work, the outputs of these clustering methods serve as a visual tool for data integration purposes. The classified documents are in reality databases entities (columns) extracted from relational databases into files. In other words, every input document is a column in a relational database model. These documents are classified based on their content in order to join or integrate two or more relational databases.

The majority of the current tools are ontology based which means that the schema matching is done manually. Manually specifying schema matches is a tedious, time-consuming, error-prone, and therefore expensive process [10]. This section describes why semantic based data integration is relevant for schema mapping of heterogeneous data. It will be demonstrated in detail which type of heterogeneities the semantic based data integration can solve.

2.2.1 Semantic integration problem (semantic conflicts)

Some of the current work in data integration research study the semantic integration problem which is different from the ontology based data integration technique. This problem is not about how to structure the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources. For example, if two companies merge their databases, certain concepts and definitions in their respective schemas like "earnings" inevitably have different meanings. In one database it may mean profits in dollars (a floating point number), while in the other it might be the number of sales (an integer). In order to adequately integrate these two columns despite the semantic integration problem, two main techniques might be used: *ontology based data integration* versus *semantic based data integration*. As mentioned previously, the first one emphasizes the database structural model while the second one focuses more on the semantic content of database entities. Both techniques are detailed in the following two sub-sections.

2.2.2 Ontology based data integration

A common strategy for solving the problems described below is the usage of ontologies which explicitly define schema terms and thus help to resolve semantic conflicts. This approach is also called ontology based data integration [14].

In recent years, data integration became very popular as a result of technological development, in consequence of which, many existing schema mapper systems were developed such as:

- BEA WebLogic Workshop [15]
- IBM WebSphere [16]
- TibCo BusinessWorks [17]
- Altova [18]
- Stylus Studio [19]
- Cape Clear [20]
- Sonic Software [21]

- ActiveState [22].

However, all these software are ontology based; therefore, they use hierarchy or network display which is a source for several problems. When the amount of data is large, this kind of display (hierarchy or network) makes the data browsing for exploration purposes very complicated. In addition, it is extremely time consuming to detect semantic content based relations between entities using ontologies because of the huge amount of data, the semantic conflict, and the unavailability of documentation on the database schemas [23] [24].

In brief, once these similarities between databases entities are detected, ontology based data integration techniques are appropriate for executing schema mapping construction operations. However, before getting there, some other techniques should be used to automatically discover these semantic similarities between database entities rather than performing this explorative task manually.

2.2.3 Semantic based data integration

To complete data integration, it is required to find the relevant information and the related entities in order to match them. For these purposes, a semantic data integration based approach is more appropriate because of the logical sequence of data integration process. First, one needs to find the entities having similar semantic content, only then can the schema mapping can be performed. As mentioned above, all the existing schema mapper systems focus on the schema mapping task itself only; therefore, the developers must perform the time-consuming task of manually finding the related entities by semantic content [23].

The manual exploration of data is hard using hierarchy or network display when the amount of data is large because it is not possible to have a global picture or macro view of the map. That is why a map is much more appropriate: it provides a global view of the database entities and allows zooming in of the interested parts.

In addition, it extremely complicated to find semantically related entities manually in such voluminous amount of data. The developer spends much time creating and editing these mappings [23] because he has to find entities having similar data content based on ontologies which can have heterogeneous meanings. For example, as discussed above a column "earnings" can have two different meanings in two heterogeneous databases: profit for one database and sales for the second one. In brief, the developer needs to have a deep understanding of the database in order to be able to create the original mapping, which is made a complicated task because of the large data size and the semantic conflicts between heterogeneous data sources.

The usage of SOM to classify the different entities, on a unified map, based on the content can save considerable time for the developer. It organizes the heterogeneous database entities from heterogeneous data sources on one single map based on their content. In addition, these entities can have heterogeneous data types on the same map. It provides the user a unified view all the database entities regardless of their data types. Furthermore, the SOM map topology reflects the similarity between the database entities. If some entities are on the same node, then it means that there is a high probability that these entities have similar semantic content. If they are located on different nodes but in the same neighborhood, they most likely do not have highly similar content, but there are still similarities to some degree. More details on the SOM based topology can be found in the Chapter 4. In brief, at this point the developer is able to *easily* find the related entities based on the content; however he still needs to validate these relations by verifying the content and ontologies.

2.2.4 Benefits of the SOM for solving data integration heterogeneities

As explained previously, research in schema matching seeks to provide automated support to the process of finding semantic matches between two schemas. However, this process is made harder due to heterogeneities at the following levels [25]: syntactic heterogeneity, schematic or structural heterogeneity, representational heterogeneity, and

semantic heterogeneity. The proposed SOM based visual data integration tool treats all these issues as explained in the next subsections.

Syntactic heterogeneity

Syntactic heterogeneity comprises all aspects that are related to the specific technical choices for the representation of interfaces and data. For example, the differences in the language used for representing the data elements are an important point. Because the data sources are heterogeneous, located at several locations and possibly managed by different groups that use different technologies, the exploration of all these data is challenging because of the dissimilarities in the technologies used.

A simple a solution to this problem is to just transform all the columns from the different unfamiliar databases into data files of the same type, e.g. text files, and then apply SOM algorithm to statistically classify them based on the content. SOM is an unsupervised algorithm which means no information is needed on the existing database schema nor on the expected matched entities (classes) in order to classify these syntactically heterogeneous entities. The final result would be a unified map topology of all these entities that reflects their relationships based on the content, regardless of their syntactic heterogeneity.

Schematic or structural heterogeneity

Schematic or structural heterogeneity means differences in the types or structures of the elements. In this thesis we focus on the extraction of a richer coincident classification by mining numerical versus textual **types**. This heterogeneity in the data types makes their mining complex. There are other data types as well such as multimedia types; however dealing with the structural heterogeneity of just these two data types and performing coincident clusters extraction from them has its value and significance. Such an extraction allows a unified view of all such data. It also allows us to extract better classification results from heterogeneous textual and numerical data types. Finally, it is the first step towards mining heterogeneous data mining of other data types.

Our proposed solution to this problem of simultaneously mining heterogeneous data is to process them by unified vectorization. In this thesis, using self-organizing maps is used because of its practical visualization features. The numerical and textual data are processed simultaneously which results in a unified semantic map of all the data despite their heterogeneous schematics. It is an interesting advancement in information retrieval and machine learning fields because the extracted information are richer, the processing natural, and the user has an unified view of all the database entities or documents despite their structural heterogeneity.

The information extracted is richer because the combination of the information extracted from the two data types (textual and numerical) leads to better clustering results. It is more natural because the two data types are processed simultaneously and organized based on their content on the same unified map. Other research projects resulted in dissimilar clusters from structurally heterogeneous data because the two data types are processed separately which causes the combination of the results to be divergent [1], [2], [3]. The resulting map is composed of textual, numerical, and hybrid textual and numerical files. In other words, a unique topology shows the user a semantic classification of all the database entities regardless of their structure or type.

Model / Representational heterogeneity

Model or representational heterogeneity means the differences in the underlying models (database, ontologies) or their representations (relational, object-oriented, RDF, OWL).

Similar to the syntactic heterogeneity problem, this difficulty is resolved by first transforming all the database entities from different source files into unstructured files of the same type, e.g., text files. These files are then processed by an unsupervised algorithm, in this case SOM, to extract their content similarities. The resulting map facilitates the user to visually explore relationships among entities despite their difference of models or representations. A portion of the original model is shown in the labels: column@table. However, an abstraction is made on the respective database models which are classified purely based on their content.

Semantic heterogeneity

Semantic heterogeneity is where the same real world entity is represented using different terms. For example, "customers" in one database may be named "clients" in another, yet the information represented by these database entities is similar.

Salton [26] and van Rijsbergen [27] were the first to introduce textual IR techniques for classifying textual database based on the content. In this paper we extended their approach by proposing a new measure of numerical data, and combine the processing of the both numerical and textual data by unified vectorization. This way, one can maximize the usage of available heterogeneous information, which was not possible before because of the difficulty of extracting coincident clusters from heterogeneous data types. In addition, this is the only way to achieve a tight coupling, since information on the data schemas is often not available because they reside at different locations or perhaps even do not exist at all in a complete form.

Chapter 3
Pre-Processing

The data does not necessarily have to be pre-processed at all. However, in most real tasks pre-processing is important; it may even be the most important part of the whole process [7]. In data integration context, pre-processing phase is important because of the huge size of the database that makes them highly susceptible to containing noisy, missing, and inconsistent data [28].

There are several pre-processing techniques used in data mining, the best-known ones are *data cleaning*, *data integration*, *data transformation* and *data reduction* [28]. *Data cleaning* is used to correct the incomplete (e.g. missing values), noisy (containing errors) or inconsistent (having discrepancies) data. *Data Integration* or *schema re-consolidation* serves to merge data from data sources into coherent data stores such as data warehouses; the consolidate data can cause inconsistencies and redundancies. *Data transformation* enhances the data mining by using for example *normalization* to prevent features with large range like "salary" from outweighing those with smaller ranges like "age". Finally, *data reduction* is used to reduce a big dataset into smaller representation that can undergo similarity analysis and is easier to process.

The main objective of this thesis is to develop a semi-automatic tool for data Integration. All the pre-processing techniques we use are shown in Figure 1. We propose three new techniques: BF-IDBF measure for numerical data, processing by unified vectorization, and VSM matrix cleaning which was mainly used in the post-processing phase.

Figure 1 The used data pre-processing techniques

3.1 Data cleaning

As the name indicates, data cleaning routines clean the data by filling the missing or incomplete values, detecting or removing the data containing errors or outliers, and resolving the inconsistencies in the data that contain discrepancies. However in the present context, the main focus is elimination of the noise in the data and improvement in the representation of the data heterogeneous, with the ultimate aim of better quality in classification.

Usually, detecting the noisy data can be done using histograms, cluster analysis or regression functions [28]. We have experimented with using several data cleaning techniques relevant to textual and numerical data. Our objective is, through data cleaning techniques, to find a way to simultaneously process heterogeneous data types by using SOM and extract coincident classification based on their content.

We begin by describing the document representations that were used, also by means of data cleaning techniques: TF-IDF for texts and histogram for numerical data. The textual representation serves as a reference to the numerical data representation because numbers are text but not the reverse. In turn, we define a new measure of numerical data, bin frequency inverse bin frequency (BF-IDBF), which represents numerical data in a very similar way as TF-IDF rather the classical histograms. Thus, it enhances the classification results for processing by unified vectorization.

3.1.1 Data representation

In order to implement any classification technique, it is necessary to transform the input documents into an algebraic model so they can be processed. The standard practice in information retrieval [29] is to represent documents as vectors in *t*-dimensional Euclidean space where each dimension corresponds to a word (term) of the vocabulary [26]. Despite its simplicity and efficiency, the vector space model (VSM) has the limitation of focusing only on textual data types, and it is hard to obtain accurate information of semantic relatedness automatically from textual information only [30]. Therefore, we propose to

use textual data representation as reference and combine numerical data to it for processing by unified vectorization.

In this thesis, the documents are database relational entities. In other words, every document is a column extracted from relational database table. In some literature, the term *field* is used to refer to a column. To illustrate how documents are obtained, consider the following example. Suppose a table named **student** represents students, and it has the following columns: id, name, dateBirth, *etc*. The documents extracted, assuming the naming convention filename.dat, will be: student@id.dat, student@name.dat, student@dateBirth.dat, *etc*.

Several data types in the databases are available to enhance data mining results; however, processing two heterogeneous data types is already very complex. Some business intelligence related project tried to extract concurrent clustering from textual and numerical data, using SOM, but the results were divergent [2], [3]. This divergence is caused by the complexity of combining quantitative and qualitative mining which led to poorer classification results.

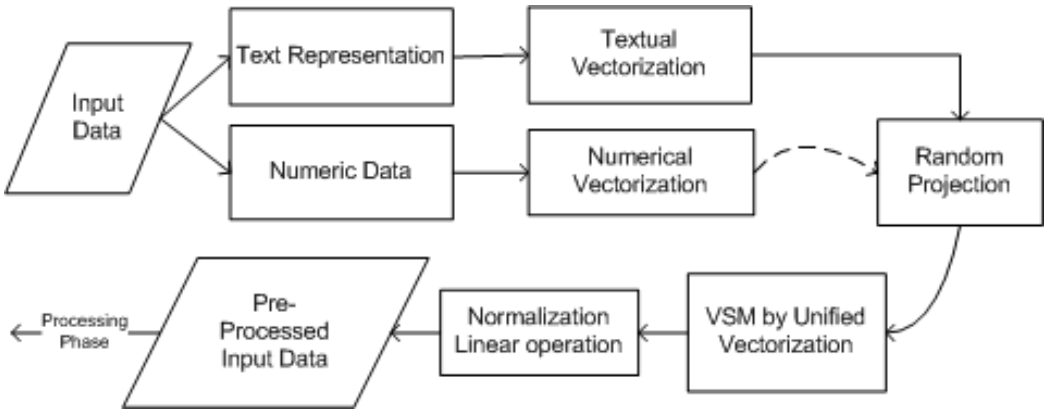


Figure 2 Proposed pre-processing steps

Therefore, our proposed approach is to pre-process heterogeneous data types (*i.e.*, numerical and textual) separately. Then as shown in Figure 2, combine their representations by unified vectorization in order to processes them *simultaneously* using

SOM algorithm. This bypasses the difficult task of combining the heterogeneous data processing outputs and has the potential to obtain better co-occurring semantic clustering results.

3.1.2 Text tokenization

The vector space model (VSM) is the most commonly used approach to represent textual data: a word is denoted by a numerical vector obtained by counting the most relevant lexical elements present in the textual portion of the document.

All textual portions of the columns d_i are transformed into a vector:

$$d_i = (w_{j1}, w_{j2}, \dots, w_{jn})$$

Where n is the number of terms in the whole set of terms (or descriptors) T, and w_{kj} represents the the weight of the term t_k in the document d_j.

Finally, the VSM matrix, also called document-term matrix, is built by putting together all the document vectors as shown in Table 1.

Docs	**Terms**			
	t_1	t_2	...	t_n
d_1	w_{11}	w_{12}	...	w_{1n}
d_2	w_{21}	w_{22}	...	w_{2n}
...	...	...	...	...
d_m	w_{m1}	w_{m2}	...	w_{mn}

Table 1 VSM matrix for textual data

Bag of words

The most common text representation within VSM framework is the "bag of words" [31][29]; it is simply the transformation of the texts portion into vectors where every term

is actually a word from the set of terms *T* [32]. In brief, this representation is straightforward and fast to implement, which is why it is so popular. The problem with this representation is that it does not consider the grammatical structure, nor the etymological analysis of the texts.

Example:

Suppose we have a document with the sentence: “Hello world”.

Simply, two terms will be extracted: **hello** and **world**

It should be mentioned that all the words are transformed into lower case in order to avoid duplicating a word because it contains capital letters.

N-Gram

One of the issues with bag of words is that it does not consider grammatical or semantic aspects. In contrast with classical textual mining, for our problem of processing database entities, the grammatical analysis is not that relevant, because each column field usually does not represent sentences but rather names, addresses, phone numbers, *etc*. On the other hand, databases contain a lot of noise in the data because of spelling errors and the absence of correctors when entering data can exacerbate that, consequently, semantic treatment is very important.

Stemming [33] is an alternative that alleviates the above-mentioned problem by looking into the morphological roots of words. For example, a stemming algorithm reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish". *Lemmatization* [34] could be another suggested approach for these syntax analysis related issues because it looks for the lemma of the word based on the context. For instance, the word "better" has "good" as its lemma, but this is missed by stemming. However, these techniques do not solve the problem related to misspelling for example, they are complex to implement, and the processing time is longer.

In order to solve these problems, we propose another method of representation based on N-gram that is commonly used in text mining. N-gram is a substring of *n* consecutive

characters generated for a given document by mainly displacing a window of n characters along the text [35]. It offers several advantages: it is an easy and fast way to solve the syntax problems mentioned earlier and detect the noise caused by misspelling and outliers. Also, it finds common patterns between words with the same roots but different morphological forms (e.g., finance and financial), without treating them as equal, which happens with word stemming [36]. Additionally, it offers better clustering performance with the lowest dimensionality than word or term representation can offer [36].

Example:

Suppose that we have a document with the phrase: “Financial group”.
When tokenized into 3-gram terms, it gives: fin, ina, nan, anc, nci, cia, ial, gro, rou, oup.
After alphabetical based reordering, it gives: anc, cia, fin, ina, gro, ial, nan, nci, oup, rou.

Assume now another phrase: “groups of finances”
After 3-gram tokenization, we get these terms: gro, rou, oup, ups, fin, ina, nan, anc, nce, ces.
When reorganized alphabetically, it gives: anc, ces , fin, gro, ina, nan, nce, oup, rou, ups.

When the 3-gram tokenizations of the two sentences are compared, there are 6 common terms out of 10(shown in bold below). It reflects the similar content between the two different phrases. In addition, the algorithm is very simple, consequently, easy to implement and fast because it is less demanding on machine processing resources.

anc, cia, **fin**, **ina**, **gro**, ial, **nan**, nci, **oup**, rou
anc, **ces** , **fin**, **gro**, **ina**, **nan**, nce, **oup**, rou, ups

Vectorization of textual data

After the corpus terms (tokens) determination through the different tokenization techniques described earlier, their respective weights need to be vectorized in order to construct the VSM matrix described earlier.

Term Frequency-Inverse Document Frequency (TF-IDF) weighting

Most textual vectorization approaches [31] [37] are based on a vectorial representation of texts using the term frequency - inverse document frequency (TF-IDF) measure. The TFIDF function weights each vector component (each of them relating to a word of the vocabulary) of each document as follows:

$$\text{TF IDF}(t_k, d_j) = \frac{Freq(t_k,\ d_j\ )}{\sum_k Freq(t_k,\ d_j\ )} \times Log \frac{\text{N}_{\text{doc}}}{\text{N}_{\text{doc}}\ (\text{t}_\text{k})} \geq 0$$

where $Freq(t_k,\ d_j\ )$ denotes the number of times the term t_k occurs in the document (column) d_j, $\sum_k Freq(t_k,\ d_j\ )$ is the occurrences total number of all the terms in the same document d_j, and N_{doc} is the total number of documents in the corpus, while $N_{doc}\ (t_k)$ is the number of documents, in the corpus, with the term t_k.

In reality the Term Frequency (TF), gives a measure of the importance of the term t_k within the particular document d_j. Thus, the more a word appears in a document (e.g., its TF, term frequency is high) the more it is estimated to be significant in this document. The term frequency is defined as follows:

$$\text{TF}\ (t_k, d_j) = \frac{Freq(t_k,\ d_j\ )}{\sum_k Freq(t_k,\ d_j\ )} \leq 1$$

Example: suppose that we a document d_1 contains 30 words. Among them the word "neurology" is repeated 4 times. Therefore the TF (neurology, d_1) = 4/30 = 0.13.

The inverse document frequency (IDF) is a value of the importance of the term in the corpus. This value is estimated using the whole training text collection at hand. Accordingly, if a word is very frequent in the text collection, it is not considered to be particularly representative of this document because it occurs in most documents. For instance, stop words such as "the, a, an" would get very low scores. In contrast, if the word is rare in the text collection, it is believed to be very relevant for the document. Therefore, the second portion of the formulas is:

$$\mathrm{IDF}(t_k, d_j) = Log \frac{\mathrm{N_{doc}}}{\mathrm{N_{doc}\ (t_k)}}$$

The 'log' in this formula has a smoothing effect. We now consider an example where we have a collection of 1000 documents.

Example:

In order to illustrate the TF-IDF calculation and the smoothing effect of the function log, suppose one of the documents has 4 words, where one of these terms is repeated twice. For example, the document contains this sentence: "Very very nice weather". To compare the results, the TF-IDF is calculated using the log function and without using it as shown in the Table 2, for two different scenarios.

Scenarios	TF-IDF with log	TF-IDF without log
Out of 1000, there is 1 document only with the word "very".	(2/4) * log (1000 / 1) = 0.5 * **3** = **1.5**	(2/4) * (1000 / 1) = 0.5 * **1000** = **500**
Out of 1000, there are 100 documents with the word "very".	(2/4) * log (1000 / 100) = 0.5 * **1** = **0.5**	(2/4) * (1000 / 100) = 0.5***10** =**5**

Table 2 Examples of TF-IDF calculation

This example shows how the log function avoids the rarity of a word in a document to outweigh the VSM matrix. In the first scenario, the TF-IDF passes from 500 to 1.5. In addition, the example clarifies how the IDF portion for the formulas reflects the rarity of token in the corpus.

Numerical data representation

Because of its different nature, the numerical data is pre-processed differently than the textual data. As an illustration, suppose we have two numbers 1988 and 1991, representing years of birth or financial values, present in two columns (documents). Their proximity will not be detected by using either word representation or n-gram because they do not possess enough textual similarities. That is why it is essential to pre-process

the numeric input data differently so that their representation reflects their semantic proximity.

Several data cleaning techniques can be used to specify concept hierarchies for numerical attributes such as binning, histogram analysis, entropy-based discretization, Z2-merging, cluster analysis and discretization by intuitive partitioning [28]. In some financial related work, pre-processing for SOM was done using histogram equalization: a method for mapping rare events to a small part of the data range, and spreading out frequent events [38]. Other Business Intelligence projects, very similar to our work, tried to process heterogeneous textual (qualitative) and numerical (quantitative) data, but the results were divergent [3] [2]. In the first such previous project [3], histograms were used to pre-process *textual* data. In the second one [2], they standardized the numerical data by scaling the variables according to the variance.

The output of these experiments is that every data should be first of all represented in a natural way. For example, textual data are usually represented by bag of words or n-grams. For optimal results their representation should not be changed or at least should be similar to their traditional way of representation. Therefore, representing textual data by histogram, which is traditionally used for numerical data, is not appropriate.

Histogram

In this research, histogram analysis is used to ease the SOM neural network's learning process and improve the quality of the map. Histogram is an unsupervised discretization technique because it does not use class information. It partitions the numerical values of a document, d_j from the corpus into disjoint ranges called buckets or bins.

In particular, *Equal-Frequency (Equal-Depth)* histogram is used in this thesis because of its good scaling properties and simplicity in implementation. The values are partitioned so that, ideally, each partition contains identical number of tuples [28]. Another very good reason for using histogram is to reduce dimensionality of VSM by at least s times, where s is the size of the bin. However, this being said, the size of the bins can be bigger than s in order to avoid cutting a cluster of the same value in the middle.

The equal-depth histogram is built by extracting all the numerical data n_i of the document d_j transforming them into a vector:

$$n_i = (v_{1j}, v_{2j}, \dots, v_{|N|j}),$$

where, N is the total number of histogram bins, and v_{lj} represents the number of observations that fall into disjoint bin b_l.

Finally the VSM matrix is built by putting together all the document vectors n_i as shown in Table 3.

Docs	**Bins**			
	b_1	**b_2**	**…**	**b_n**
d_1	v_{11}	v_{12}	…	v_{l1}
d_2	v_{21}	v_{22}	…	v_{l2}
…	…	…	…	…
d_m	v_{m1}	v_{m2}	…	v_{mk}

Table 3 Numerical portion of the VSM matrix

Example:

Let us assume that we have the following group of numbers in the corpus which are ordered: 0.5, 1, 5, 7, 7, 9, 17, 2000, 3500, 50000. Suppose the bin size is 4.

b_1 = {0.5, 1, 5, 7, 7} → The bin has 5 numbers in order to not cut the cluster and separate the number 7 into two different buckets.

b_2 = {9, 17, 2000, 3500}

b_3 = {50000}

BF-IDBF

After trying the histogram weighting measure for the numerical data and processing together with the textual data by unified vectorization, it appeared that the results were good but not optimal [39]. The detailed results can be found in the case study (sub-section 6.4). In brief, the results when using the bag of words are substantially better than

using N-Gram tokenization despite the normalization of the two measures before processing. The reason is that TF-IDF weighting for textual data does not have the same meaning or representation as the histogram. More precisely, the histogram measure does not reflect the weight of the numerical term within the corpus as the IDF portion does. Therefore, this difference in the results shows how important the influence of the pre-processing phase can be on the final processing results. Consequently, when unified vectorization is used for processing heterogeneous data types, their respective vectorization measures or weights should represent the same kind of information.

Bin frequency (BF) measure

As a solution to the representation difference between TF-IDF and histogram measures, the Term Frequency-Inverse Document Frequency (BF-IDBF) weighting for numerical data is proposed as similar representation to the textual TF-IDF data representation [40]. The BF is calculated in a similar way as TF; it serves to estimate the importance of bin rather than the relevance of the number in a document. In other words, both measures, the histogram and TF, are combined to form the BF. The BF is estimated as follows:

$$\mathrm{BF}(b_l, d_j) = \frac{Freq(b_l,\ d_j\ )}{\sum_k Freq(b_l,\ d_j\ )},$$

where $Freq(b_l,\ d_j\ )$ denotes the number of times the bin b_l occurs in the document (column) d_j, $\sum_k Freq(b_l,\ d_j\ )$ is the total number of all the bin occurrences in the same document d_j.

Other variances of histogram could be used as well, but because of the good results obtained in our previous work [39], “equal depth" histogram was kept.

Inverse bin document frequency (IDBF) measure

After calculating the BF, the next step is the calculation of the IDBF weight which mainly serves to reduce the weight of the bins that are insignificant. In other words, if a number or certain range of numbers is common to a large number of documents, the

weight is decreased for better document classification based on the numerical semantic content. The IDBF is approximated through a similar formula to IDF calculation:

$$\text{IDBF}(b_l, d_j) = Log \frac{\text{N}_{\text{doc}}}{\text{N}_{\text{doc}}\ (b_l)},$$

where N_{doc} is the total number of documents in the corpus, while $N_{doc}\ (b_l)$ is the number of documents in the corpus with bin b_l.

Combination of BF and IDBF

Finally, the BF-IDBF is calculated by multiplying the two measures; therefore, the global formula is:

$$\text{BF IDBF}(b_l, d_j) = \frac{Freq(b_l,\ d_j\ )}{\sum_k Freq(b_l,\ d_j\ )} \times Log \frac{\text{N}_{\text{doc}}}{\text{N}_{\text{doc}}\ (b_l)}$$

As explained previously when illustrating the usage of TF-IDF, the log serves to smoothen the results.

Example:

Suppose there is a series of numbers, which have been ordered, in a corpus C: 0.5, 1, 5, 7, 7, 9, 17, 2000, 3500, 50000…

Let us have a bin size of 4.

b_1 = {0.5, 1, 5, 7, 7}, b_2 = {9, 17, 2000, 3500}, b_3 = {50000,…},…, b_n.

In this example b_1 has 5 elements because, the number 7 should not be in two different bins.

Assume that the document d_1 contains 15 numbers which are distributed among 9 bins. Among these 15 numbers there are these three: {0.5, 1, 5}.

Suppose that in a corpus of 1000 documents, there are two documents each of which contains one instance of the number 7. In other words, the bin b_1 can be found in only three documents: d_1, d_2, d_3.

$C = d_{1, \ldots,} d_{1000}$

$\{0.5, 1, 5\} \in d_1$

{7}∈ d_2

{7}∈ d_3

Therefore; the BF-IDBF of the bin b_1 in the document d_1 is calculated as follows:
BI-IDBF (d_1, b_1) = (3/9) * log (1000/3) = 0.33*2.52 =0.84

3.1.3 Data transformation

Normalization

Before processing the data, the data should be normalized in order to avoid an unjustified influence of one of the two data types (textual and numerical) during the SOM training phase.

There are many methods for data normalization; the most commonly used in data mining are: min-max normalization, z-score normalization, and normalization by decimal scaling [28]. In the literature, it was found that normalization according to the variance for example was used to pre-process the financial data for benchmarking using self-organizing map [41].

In this thesis min-max normalization was applied because of its simplicity and efficiency: a linear transformation of the original input range into a newly specified data range, typically [0, 1].

$$y' = \frac{y - min}{max - min} (max' - min') + min',$$

where the old min value is mapped to new *min*: *min'*. The Old max is mapped to new *max*: *max'*. Let *y* be the original value, *y'* be the new value. The min and max are the original *min* and *max* values. The *min'*, *max'* are the new *min* and *max*.

All the values of the unified VSM matrix were normalized similarly in a range of [0, 1] through this linear operation.

Combination of textual and numerical mining by unified vectorization

After passing the heterogeneous textual and numerical data through the different pre-processing phases (see Chapter 3 for more details) - data cleaning, data transformation and data reduction - we propose to use the unified vectorization method of numerical and textual data in order to process them simultaneously, and meet the challenge of a extracting better classification and mining results from these heterogeneous data.

The data is simply put together as shown in Table 4, and then re-normalized a second time, as described in the previous section, in order to have an equivalent weight for the numerical and textual data.

Docs	**Terms**				**Bins**			
	t_1	**t_2**	**…**	**t_n**	**b_1**	**b_2**	**…**	**b_n**
d_1	w_{11}	w_{12}	…	w_{1n}	v_{11}	v_{12}	…	v_{l1}
d_2	w_{21}	w_{22}	…	w_{2n}	v_{21}	v_{22}	…	v_{l2}
…	…	…	…	…	…	…	…	…
d_m	w_{m1}	w_{m2}	…	w_{mn}	v_{m1}	v_{m2}	…	v_{mk}

Table 4 Unified vectorization of textual and numerical data

Why combining heterogeneous data by unified vectorization for processing?

Unified vectorization is proposed for a couple of reasons. First, as mentioned above, other researchers tried to process the data types separately, then combine the outcomes which resulted in divergent results. This divergence is caused probably by the complexity of combining heterogeneous data mining results. Another aspect is that the trained SOM map does not always result in the exact same topology after every training. It may differ as a consequence of the random initialization of the SOM map. Therefore, combining the textual and numerical SOM's resulting topologies is a complex task.

In order to resolve these issues, we propose to use the unified vectorization technique for processing heterogeneous textual and numerical data. Consequently, the training will be faster because the heterogeneous data are processed simultaneously rather than processing them in sequence by data types. Moreover, the challenging task of combining

the results is avoided, and naturally the same trained SOM's map is used to classify and visualize these heterogeneous data types.

3.1.4 Dimensionality reduction

In a number of cases, the *high dimensionality* of the data leads to cumbersome computations and even restricts the choice of data processing methods. In text mining context for example, the high dimensionality is due to the large vocabulary. Therefore, the data dimension must be reduced by preserving the information for data mining purposes. A statistical optimal of dimensionality reduction is to project the data onto a lower-dimensional orthogonal subspace that captures as much of the variation of the data as possible.

The most widely used way to do this is principal component analysis (PCA); it is known to give good results and has a lot of useful properties. However, it is computationally expensive and is not feasible on large, high dimensional data [42].

Random Projection

Another powerful technique to reduce data dimension is the Random Projection (RP) which is simple, offers clear computational advantages and preserves similarity [42]. RP was successfully tested with SOM on several applications using textual data and images. It appeared to be a good alternative to traditional methods of dimensionality reduction that are computationally infeasible for high dimensional data as opposed to RP which does not suffer from the curse of dimensionality [43]. It was shown that random projections are best suited for use with Nearest Neighbor methods and they also combine well with SVM [42].

Another good reason for using random projection with textual application is that is useful in query matching if the query is long, or if a set of similar documents instead of one particular document were searched for [43]. Also, it should be recalled that the numerical data dimensionality was reduced previously by at least s times by transforming it into histograms bins.

In brief RP works as follows: given a matrix X, the dimensionality of the data can be reduced by projecting it through the origin onto a lower-dimensional subspace, formed by a set of random vectors (R):

$$A_{[m \times k]} = X_{[m \times n]} \cdot R_{[n \times k]}$$

The variable *k* is the desired reduced dimensionality.

3.1.5 Pre-processing review

As shown in Figure 3, this section presented an efficient way to pre-process heterogeneous textual and numerical data for better coincident clustering and classification results.

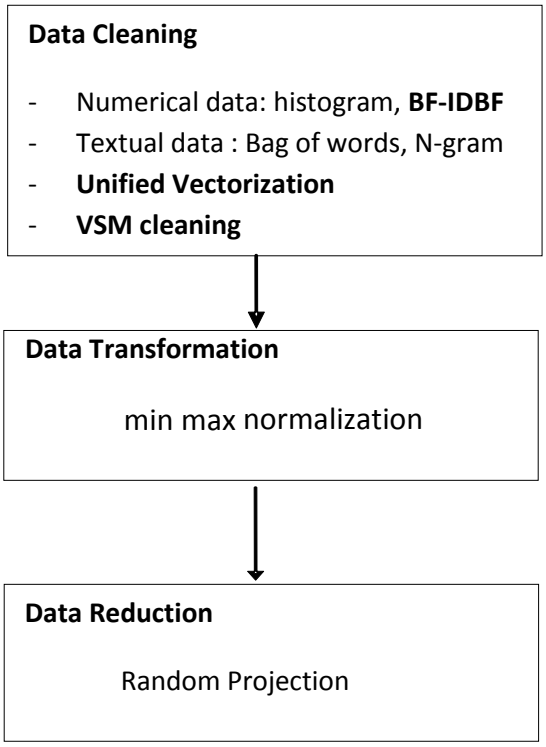


Figure 3 Pre-processing steps

Firstly, unified vectorization was introduced, and it resulted in extracting better *meaningful* results. This technique permits the processing of different data types simultaneously in a more natural way with better results. Consequently, it simplifies the

visual classification by permitting the usage the same trained SOM map for clustering heterogeneous data types rather than processing them separately, and then trying to combine the clustering outcomes which leads to poorer results because of the high complexity of the combination task.

Also, the new suggested weighting measure BF-IDBF improved significantly the efficiency of the SOM algorithm. It is likely that this measure would improve the performance of other machine learning algorithms when UV is used. Other existing techniques such as data cleaning methods, RP, min-max normalization were used to enhance the classification.

Chapter 4
Processing

4.1 Self organizing map

Self-Organizing Maps (SOMs) was developed by Kohonen in 1980's. It is an unsupervised learning method which is based on the principle of competition according to an iterative process of updates [43]. It has been extensively used in various fields.

As shown on Figure 4, the most remarkable capability of SOM is to produce a mapping of high dimensional input space onto a low-dimensional (usually two dimensional) map, where similar input data can be found on nearby regions of the map. The resulting map offers a better insight into the interrelationships among the input data and helps identify clustering tendencies.

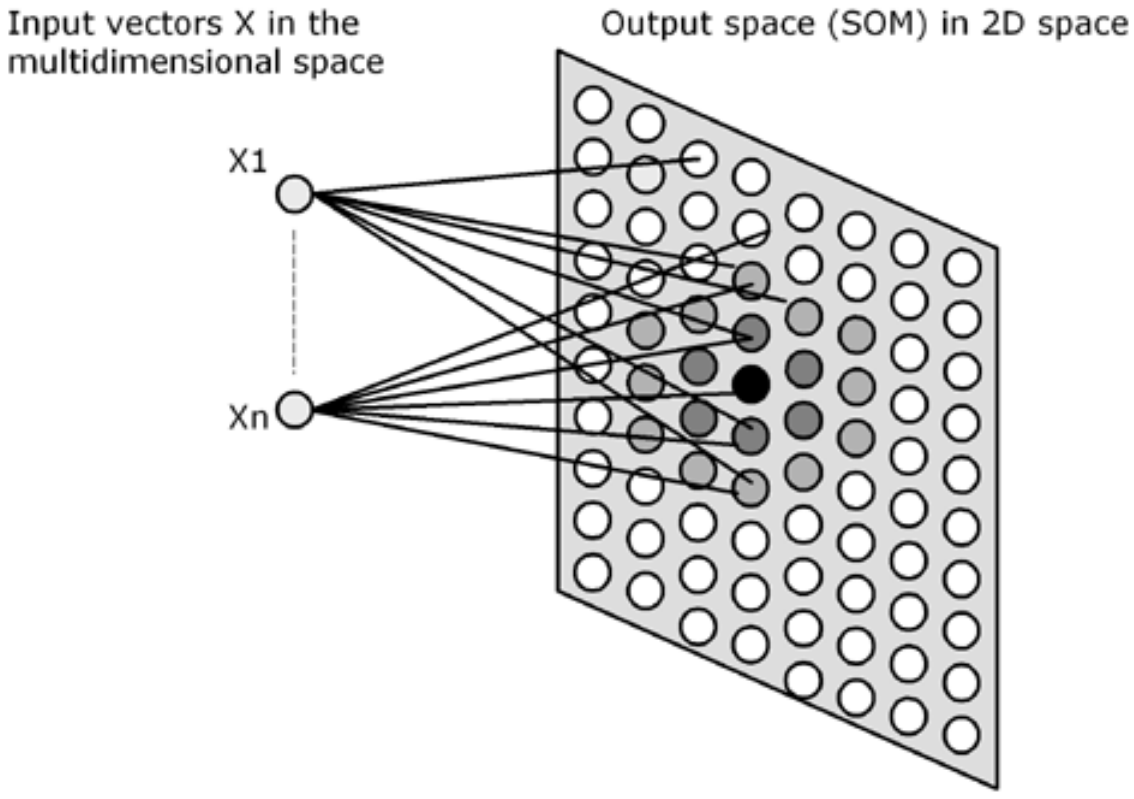


Figure 4 SOM: Projection of high demensional data into two dimensional space

Like most artificial neural networks, SOMs operate in two modes: training and mapping, which are explained in the beginning of the chapter. Later on, two versions of SOM are presented as well as the initialization. Then, the mapping and labeling processes are described.

4.2 SOM training

4.2.1 Training initialization

The number of neurons should usually be selected to be as large as possible, with the neighborhood size controlling the smoothness and generalization of the mapping. The mapping does not considerably suffer even when the number of neurons exceeds the number of input vectors, if only the neighborhood size is selected appropriately. However, as the size of the map increases, *e.g.*, to tens of thousands of neurons, the training phase becomes computationally impractical for most applications.

First, the number of map units (*M*) is determined automatically. The heuristic formula used is:

$$M = 5 * \sqrt{\mathrm{n}},$$

where *n* is the number of input vectors. Note that the computational load increases quadratically with the number of map units, so because of hardware limitation, the number of map units should be smaller than 2000.

Then the SOM is initialized. First, linear initialization along two greatest eigenvectors is tried; otherwise, a random initialization is performed.

4.2.2 Training modes

SOM has two training modes that are mentioned in the literature: sequential and batch version. They differ in the method of updating m_j weight vectors. The advantages of batch method over the sequential version are: a) it produces a map much faster and b) it does not need a learning rate to converge [45], [26] and [46].

Sequential Version of self organizing map

The goal of learning in the self-organizing map is to cause different heterogeneous data sets to respond similarly to certain input patterns.

Let $x_i \in \Re^T, 1 \leq i \leq N_{doc}$ be the feature vector of the i^{th} document in the corpus, where T is the number of indexed terms and N_{doc} is the number of documents. These vectors are the training inputs to the map.

The map is a regular grid of $M = m\ x\ n$ neurons (units). Similarly to the input vectors, each neuron m_j in the map has T dimension. Let $w_j = \{ w_{kj} \mid 1 \leq k \leq N_{doc} \wedge 1 \leq j \leq T\}$, be the weight of the m_j neuron in the map, where M is the number of neurons in the map. The weights of the neurons are initialized to small random values, or small pseudo-random values drawn from a normal distribution with mean zero and standard deviation. With the latter alternative, learning is much faster because the initial weights already give good approximation of SOM weights.

Step 1: Randomly select an input vector x_i, which represent documents, from the corpus

Step 2: Find on the map the closest node m_j, called the best matching unit (BMU) [47], to x_i having the smallest Euclidean distance c:

$$c = argmin\{x_i - w_j\}$$

where, w_j is the weight vector of the neuron m_j.

Step 3: For every neuron l in the neighborhood of the node j, update the synaptic weight by: $w_l(t+1) = w_l(t) + \propto (t)(x_i - w_l(t))$,

where, $\alpha(t)$ is the training gain at the time stamp t. The learning rate $\alpha(t)$ is a decreasing function of time between [0, 1].

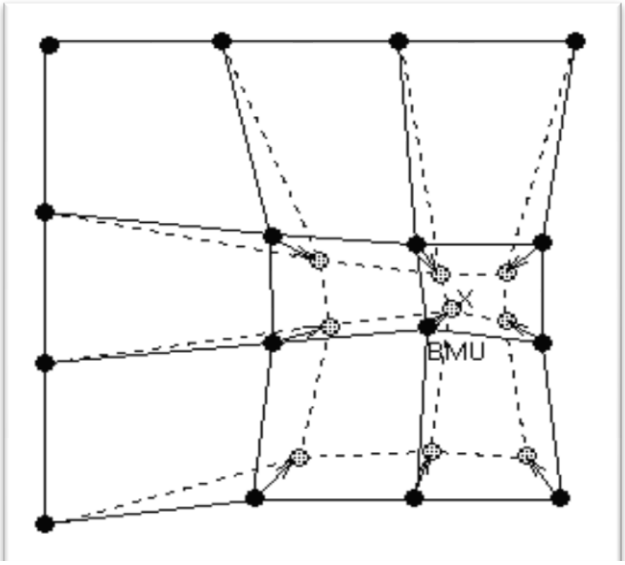


Figure 5 Updating the best matching unit (BMU)

Therefore, the topological neighbors are moved closer to the input vector in the input space as shown in Figure 5.

Step 4: Increase the time stamp *t*. If t reaches the preset maximum training time *T*, halt the training process, otherwise decrease $\propto(t)$ and go to Step 1.

The neighborhood size also shrinks as the time passes, depending on the type of the neighborhood grid, as shown on Figure 6 and Figure 7.

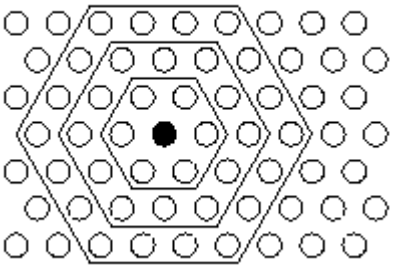


Figure 6 Hexagonal neighborhood grid

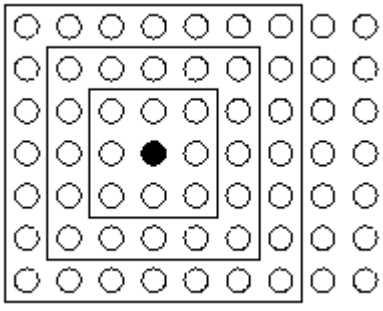


Figure 7 Rectangular neighborhood grid

The training process stops after time *T* which is large enough to pass all the input vectors as training input. More details can be found in [47].

Some articles recommend the use of hexagonal lattice because all 6 neighbors of a neuron are at the same distance (as opposed to the 8 neighbors in a rectangular lattice) and the maps become smoother and more pleasing to the eye. However, we preferred the usage of rectangular neighborhood grid because of its simplicity, and the differences in the results were insignificant.

Batch version of self organizing map

The **batch** version of SOM differs from the sequential version basically in the method of updating w_j weight vectors. Recall that the advantages of batch method over the sequential version are:

- It produces a map much faster.
- It does not need a learning rate to converge.

More details can be found in [47], [45], and [26].

In brief, the batch version of SOM works as follows:

Phase 1: Compare each input vector *x(t)* with all the map nodes m_i, initially selected randomly, in order to find the best matching unit (BMU) [47]. Then, copy each $x(t)$ into a sub-list associated with that map unit.

Phase 2: When the entire *x(t)* have been distributed into the respective sub-lists in this way, consider the neighborhood set N_i, around the map unit m_i. In other words N_i represents all the map units within the radius of map unit m_i. In the union of all sub lists in N_i, the mean of the correspondent $\bar{x}_i$ is computed, and this is done for every N_i.

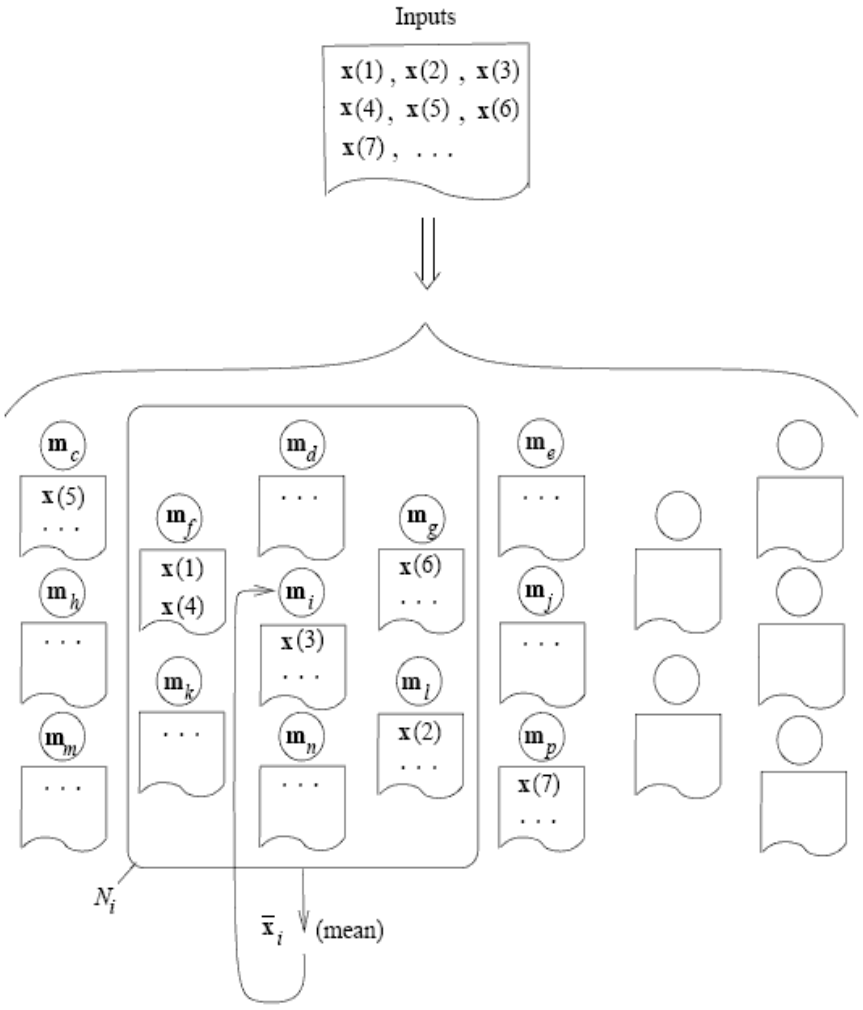


Figure 8 Batch version of SOM

Phase 3: The next step in the process is to replace each old value of m_i by the respective $\overline{x}_i$, and this replacement is done concurrently for all the m_i as illustrated on Figure 8.

$$m_i(\mathrm{t}+1) = \frac{\sum_{j=1}^{n} h_{ic(j)}\ (t)\, x_j}{\sum_{j=1}^{n} h_{ic(j)}(t)},$$

where, *c(j)* is the BMU of sample vector x_j, $h_{i,c(j)}$ the neighborhood function (the weighting factor), and *n* is the number of sample vectors.

4.3 Visualization

Zhan [46] defines the visualization as the process of transforming data, information, and knowledge into graphic presentations to support tasks such as data analysis, information exploration, information explanation, trend prediction, pattern detection, rhythm discovery, and so on. Furthermore, Information visualization aims to explore abstract data and to create new insights [48]. In data integration context, the purpose of the visualization feature is to facilitate the semantic relation exploration between database entities based on their content, and detect the similar entities that should be joined.

4.3.1 Mapping

The easiest way to map the clustered documents (columns) is to match every input vector to its respective BMU node, and this will result in having similar documents clustered on the same node, let us call that cluster Cl_k.

It is possible for a document to be part of several classes. In order to illustrate that, it is possible to map every document to its x best BMUs. However, x must respect this condition:

$$2< x \leq N_{doc},$$

where, N_{doc} is the number of input vectors (documents). In this work, only the first best matching unit was extracted.

The Figure 9 (next page) bellow is a "screen shot" of the trained map in the visualization tool that we developed. Every node (neuron) regroups a cluster of similar database entities. In this case, the *Northwind* [50] database is being processed. More details on the experimental results and database can be found in the section 6.4.

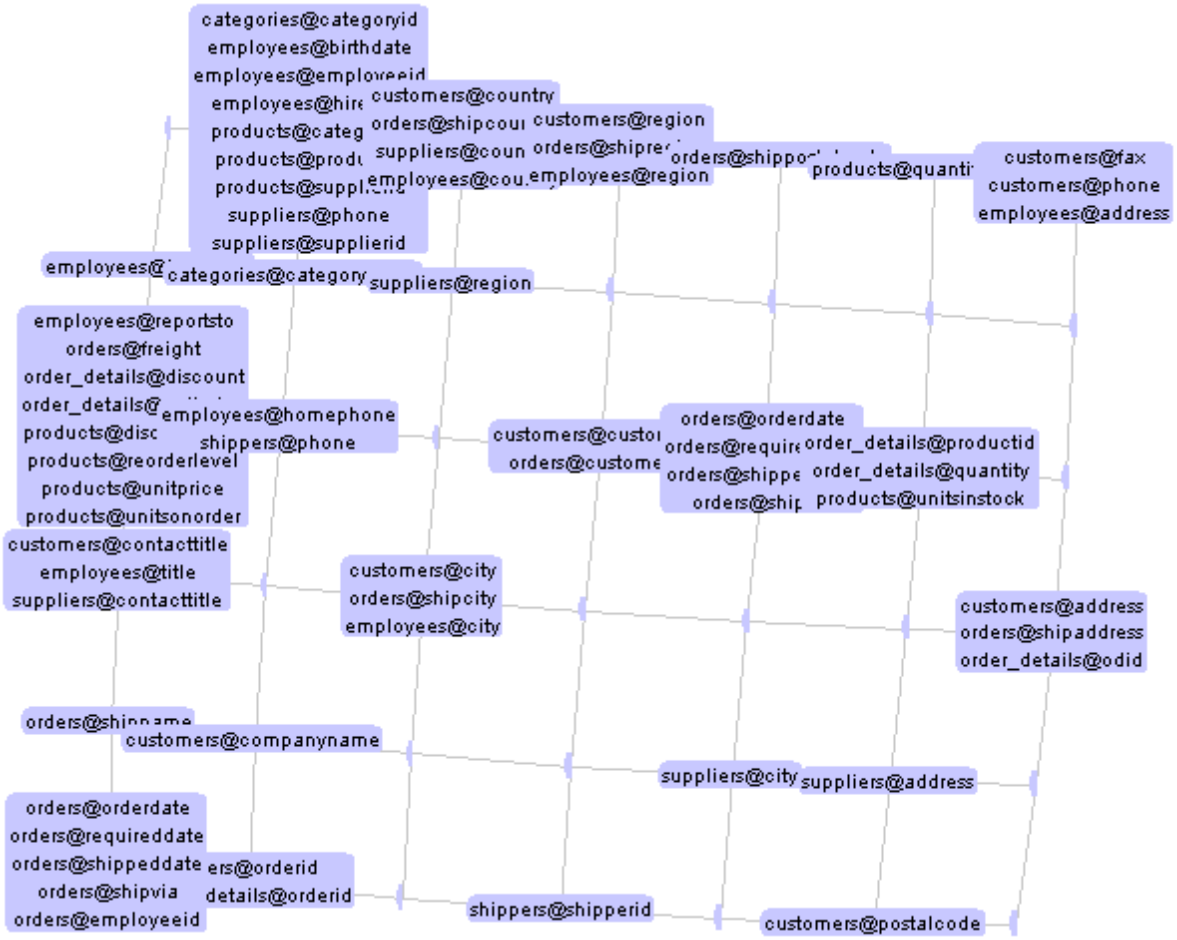


Figure 9 Trained SOM map

4.3.2 Advantage of map display

Lin proposes four type of visual display: hierarchical displays, network displays, scatter displays and map displays [51]. He then specifies that the advantages of visual display for information retrieval are characterized as:

- The ability to convey a large amount of information in a limited space
- The facilitation of browsing and the perceptual inferences on retrieval interfaces
- The potential to reveal semantic relationship of terms and documents

These advantages are demonstrated through a map display generated by neural network's self organizing map [51]. In other words, another reason of having chosen SOM is that it offers all the vantages of visual display for information retrieval.

Ability to convey a large amount of information in a limited space

In the context of schema mapping for data integration, the ability of viewing a large amount of data in a small space is very necessary. Lin [51] mentions that the geographic map is clearly the best example of using graphical displays to show large amount of information and their relationships.

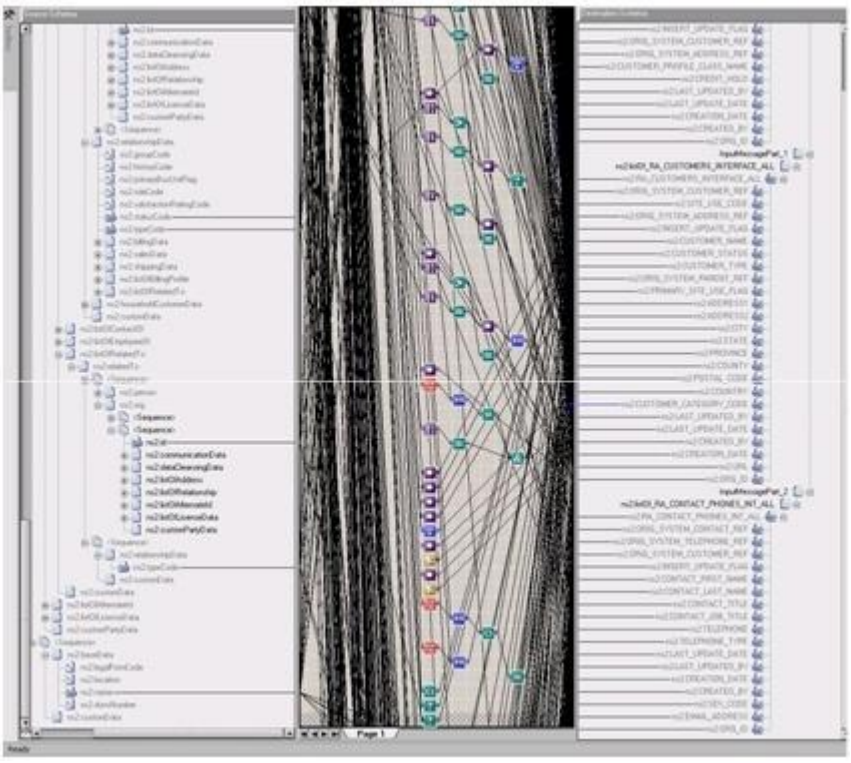


Figure 10 Example of failure to scale for large schema

The problem with the current integration tools is that they do not scale well to large schemas, and yet that is exactly what business need [23]. As shown on Figure 10, it becomes even a maze of complexity [23]. This scaling problem is caused by the fact that

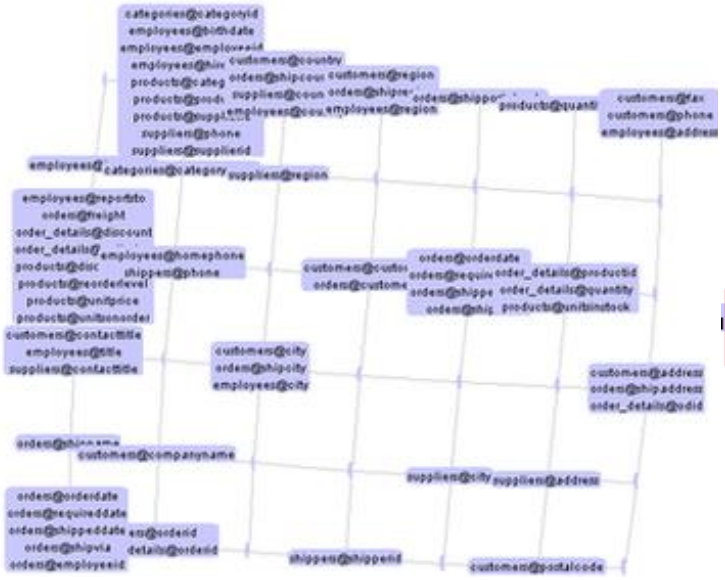
these tools use hierarchical display (tree) which is appropriate for the schema mapping task but not in large schemas because of its deficient explorative properties.

Because of these inappropriate displays in majority of the schema mapping tools, SOM is proposed as an alternative due to several visual add values. It uses a map display which offers as mentioned earlier all the vantages of visual display for information retrieval. It facilitates the semantic large schemas exploration. It permits to have **macro** detail view level similar to the cartographic maps which offer a large view of the area. That is exactly where the existing software are lacking because they use a tree display which cannot offer global picture particularly when there is a high volume of data.

Facilitation of browsing

Browsing is a direct application of human perception information seeking, both in the electronic and non-electronic environment [51]. Browsing is explorative; it is an interactive process in which one will scan large amounts of information, perceive or discover information structures or relationships, and select information items through focusing one's visual attention [51]. Browsing is an extremely important means to explore and discover information [46].

In data integration context, discovering relevant data is very important. In addition, performing joins between semantically equivalent or related entities is all the purpose of schema mapping. SOMs answer to these issues by presenting a topological map of the database entities that facilitates the user to discover relevant data, and perceive semantic relationship between entities.



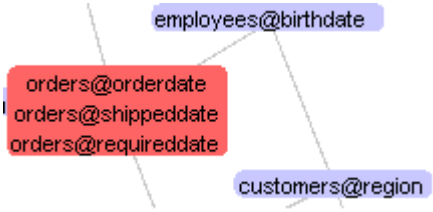


Figure 11 SOM map overview **Figure 12 SOM map zooming**

As shown on Figure 11, SOM permits to the user to discover relevant database entities, and perceive semantic relationship between them. In addition, it is possible to focus on interested portion only if needed as illustrated on Figure 12.

The potential to reveal semantic relationship

Information retrieval visualization underlies a semantic framework, elucidates relationships of concepts, illustrates holistic overview, demonstrates patterns, and facilitates interaction between systems and users [46]. Consequently, the visual tool in order to be an adequate explorative instrument must provide the user a way to discover semantic relationship between the different entities.

In schema mapping for data integration context it appears, according to the Microsoft research group based on several years of feedback from real users of schema mapping tools, that there are two primary problems [23]. First, when a developer wants to map different schemas, help is needed in understanding the semantic relationship between

elements and the semantic meaning of the mapping. This problem gets much harder as the schemas and mappings get large [23]. Secondly, basic navigation tasks (e.g., finding what schema elements are linked to what other schema elements) are very common and are seriously impaired when the schemas and mappings get large. In other words, both the navigation task and semantic editing task become harder when schemas and mappings get large. The final task does not only become larger, but, also very time consuming. However, the resulting SOM trained map topology permits to discover the semantic relationship which is a necessary task for automated schema mapping. The map nodes represent clusters of semantically similar database entities. The closer are the nodes of the map, the closer is their semantic relationship. In addition, it is possible to use dynamic coloration to facilitate the visualization of neighboring clusters and entities. For example in Figure 13, the interested cluster is in red while the neighboring clusters are in orange.

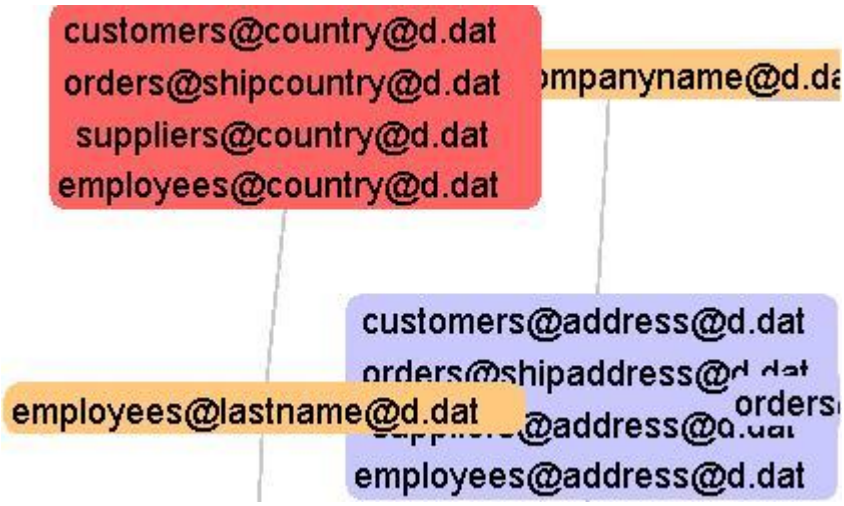


Figure 13 Coloration of the neighboring clusters.

4.4 Processing summary

This chapter presented an efficient way to integrate unfamiliar heterogeneous textual and numerical data types by the usage of SOM based visualization tool. SOMs reduce dimensions by producing a map of usually two dimensions that plot the similarities of the data by grouping the similar data items. Subsequently the map can be used for visually conveying meaningful information about the input data, exploring document similarities and perform searches. Particularly, we demonstrated how effectively simultaneous processing of heterogeneous data types can be operated for better clustering results. The resulting unified SOM trained map exposes the similarity between database entities based on their semantic content, which facilitates the semantic data exploration for database integration regardless of the data types. This tool is applicable to data integration over web data sources and tuples classification based on the content.

Chapter 5
Post-Processing

5.1 Common item-set based classifier

In the previous sections, it has been shown how to classify similar heterogeneous documents or entities based on the content. However despite the satisfactory results obtained from SOM, some issues remains when the BF-IDBF measure is not used.

Firstly, some nodes regroup several *heterogeneous* clusters together, as shown in Figure 14. Consequently, the map is unbalanced by having some nodes extremely overloaded versus some other nodes being completely empty. Secondly, some other documents are not matched to their best cluster in consequence of which the overloaded node gets even bigger, or some nodes have only one document matched to them.

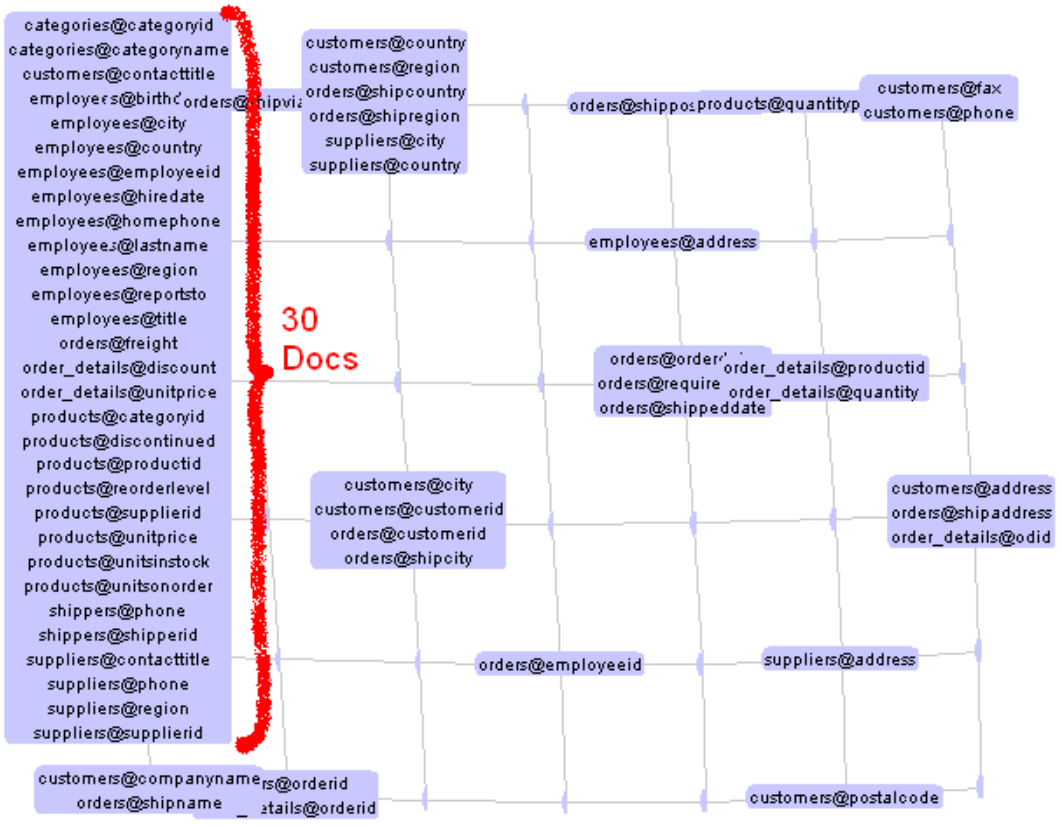


Figure 14 Unbalanced trained SOM map with dissimilar measures

Note that these unstable effects were observed when the TF-IDF vectorization measure for textual data was combined with histograms vectorization measure for numerical data portion. Probably, these outcomes are the consequence of heterogeneity of the vectorization measures because these phenomena were not observed when the TF-IDF and BF-IDBF; hence, similar representation, were used.

As a solution to the two previous unbalanced map and un-clustered documents issues, a new algorithm called *Common Item-set Based Classifier (CIBC)* is proposed in order to *smoothen* the clusters obtained in the previous section and make the visual presentation clearer. Firstly, CIBC refines the clusters by validating the homogeneity of every cluster Cl_i and re-cluster them into homogonous sub-clusters, when necessary, by preserving their topological closeness. Secondly, the algorithm finds for every un-clustered column a possible matching cluster Cl_i. Finally, it distinguishes visually on the map the clusters with homogenous data from clusters with heterogeneous data (columns) if there are any.

5.2 Algorithm description

To illustrate the CIBC algorithm, suppose that we have a normalized term-document input VSM matrix, similar to Table 5 on the next page, which has been pre-processed and then processed through the Batch version of SOM. The following step is to tune up the trained map by using CIBC algorithm in four phases: *forming item-sets*, *clusters homogeneity validation*, *clustering unique documents*, *remapping the unique documents*. These phases are described below.

5.2.1 Phase 1: Forming item-sets

Let us assume we are given a set of document items: Item-set D. Item-set $I_{t_k} \subset D$ is some subset of *similar* documents d_j (at least 2) based on the common term t_k:

$$I_{t_k} = \{d_j \in D \mid w_{jk} > 0 \wedge |I_{t_k}| \geq 2\} \qquad (1)$$

Where w_{jk} is the weight of term t_k in the document d_j from the corpus D.

At this point, some item-sets contain a high number of similar documents because of common stop words like "the" for example or some other type of noise. A simple statistical method is proposed to delete item-sets formed based on undesirable terms such as stop words. Therefore in order to eliminate these insignificant item-sets, the ratio of similar documents in every item-set $|I_{t_k}|$, based on one term, should be smaller than a certain threshold called Max_I:

$$\frac{|I_{t_k}|}{|D|} < Max_I \quad \text{where} \quad 0 < Max_I < 1 \qquad (2)$$

Example:

Suppose that $D = \{ d_1, d_2, d_3, d_4, d_5\}$ is a corpus of five documents, as modeled in Table 5. Let us assume that the corpus is composed of four terms: $\{t_1, t_2, t_3, t_4\}$.

Docs	$\mathbf{t_1}$	$\mathbf{t_2}$	$\mathbf{t_3}$	$\mathbf{t_4}$
$\mathbf{d_1}$	0.6	0.4	0	0
$\mathbf{d_2}$	0.3	0	0.4	0.3
$\mathbf{d_3}$	0.9	0.1	0	0
$\mathbf{d_4}$	0.3	0	0	0.7
$\mathbf{d_5}$	0	0	0	1

Table 5 Example term-document matrix

From the original corpus *D*, 3 item-sets are formed:

$I_{t_1} = \{d_1, d_2, d_3, d_4\}$

$I_{t_2} = \{d_1, d_3\}$

$I_{t_4} = \{d_2, d_4, d_5\}$

Let us eliminate the insignificant item-sets by using the formula (2). For example, we can fix Max_I to 0.7. It means if an Item-set is formed of 70% of the documents (entities) it

should be eliminated because they have in common probably a stop word like "the, a, an".

For example, I_{t_1} is composed of 80% of the documents; therefore, it should be deleted. Most probably these documents have in common some stop word (t_1), that is why the ratio is too high.

5.2.2 Phase 2: SOM's clusters homogeneity validation

Recall that every node of the trained SOM map represents a cluster Cl_i. As explained earlier, some cluster Cl_i obtained from SOM's visualization phase are not heterogeneous and as a consequence some nodes are overloaded by too many documents (entities), and on the contrary many other ones are completely empty. Therefore, the heterogeneity of every cluster Cl_i should be revalidated in order to redistribute these clusters into more homogenous sub-clusters, and consequently rebalance the SOM map.

First, for every node's cluster Cl_i, it has to be found all the item-sets I_{t_k} extracted in the previous phase, having in common at least two documents. In other words, the intersection of Cl_i and I_{t_k} should respect the following rule:

$$\left|Cl_i \cap I_{t_k}\right| \geq 2$$

Then, we should be keep only the intersection of the two subsets $Cl_i \cap I_{t_k}$, let us call it: I_{Cl_i,t_k}.

Let us call all the identical Item-sets I_{Cl_i,t_k}: $\boldsymbol{I_{Cl_i,n}}$

where $n \in \mathbb{N}$.

Secondly, among all the item-sets or sub-clusters $I_{Cl_i,n}$ located on the node i, we should find the largest one that will remain on the same node, while the other ones are moved to other empty nodes in the neighborhood. In other words, it should be determined which item-set $I_{Cl_i,n}$ has the largest number of documents, and let us call it for simplicity reason:

Cl'_i. However, Cl'_i should respect the following condition in order to keep only the *strongly* related documents:

$$\frac{\left|T_{Cl'_i}\right|}{\left|T_{\text{Max}(d_j, Cl'_i)}\right|} > \alpha \qquad (3)$$

Where $\left|T_{Cl'_i}\right|$ is the size of the vocabulary of the sub-cluster Cl'_i, $\left|T_{\text{Max}(d_j, Cl'_i)}\right|$ is the vocabulary size of the document d_j, which belongs to Cl'_i, having the richest vocabulary. α (Usually equal to a value close to 0.05) is the threshold to keep only the strongly related documents of the current sub-cluster Cl'_i.

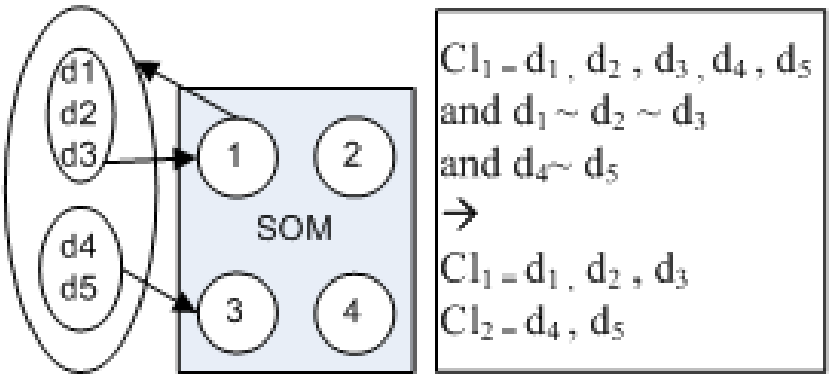


Figure 15 SOM's clusters homogeneity validation and redistribution

As illustrated on Figure 15, Cl'_1 which is the biggest sub-cluster is kept on the current node (BMU) while the remaining documents $\overline{Cl'_i}$ are re-processed until no homogeneous sub-cluster can be found on the same node. In case there is another existing sub-cluster(s) Cl''_i, it should be moved to another empty node (with no clusters) within the neighborhood. In this example, another sub-cluster Cl''_1 was found and moved to an empty node in the neighborhood: node 3.

Example:

As shown on Figure 15, let us assume the SOM's **node 1** is composed of the cluster Cl_1= {d1, d_2, d_3, d_4, d_5}.

Assume that d_1, d_2 and d_3 have a similar content and d_4 and d_5 is another group having similar content. Actually, these two clusters if separated would be the expected final result.

Let us assume that we have these items-sets: I_{t_1}= {d_1, d_2, d_3}, I_{t_2}= {d_1, d_4, d_5}, I_{t_4}= {d_1, d_2, d_3}, I_{t_5}= {d_4, d_5} and the total number of terms in the corpus is 5.

Therefore I_{t_1}, I_{t_4} are the two Item-sets having the most common items in common with the cluster Cl_1: $I_{t_1}, I_{t_4} \rightarrow I_{Cl_1,1}$ = {d_1, d_2, d_3}.

In order to create a new sub-cluster Cl'_i strong or composed of semantically strongly related entities, we have to make sure that the $I_{Cl_1,1}$, which will become Cl'_1, respects the rule (3).

$\left|T_{Cl'_1}\right| = 2$; because they have two terms in common (t_1 and t_4).

$\left|T_{\text{Max}\left(d_1, Cl'_i\right)}\right| = 5$, because per assumption the document d_1 has the five terms of the corpus.

The condition (3) is respected: $\frac{2}{5} > 0{,}05$

Therefore, Cl'_1 which is formed of {d_1, d_2, d_3} is the biggest homogeneous sub-cluster that will remain on the current node (1).

With the same logic, is determined that Cl''_1 is composed of {d_4, d_5}.

Therefore, Cl'_1 remains on the same node while Cl''_1 is moved to a neighbouring node; in this example the node 3.

5.2.3 Phase 3: Clustering unique documents

Now that the clusters are subdivided and reorganized to create more homogeneous clusters, there is an opportunity to improve the classification results by analyzing the unique entities. By unique entities, we mean the database entities that could not be matched to any existing cluster.

For every unique column (entity), the best matching cluster should be found, if it exists, respecting the phase 1. If the document is considered strongly related to one of the clusters according to the rule (3), then, it is moved to that cluster. Otherwise, the research process continues until a matching cluster is found or the clusters list expired.

As last resort, following the same process (phase 1), it should be tried to form new clusters among the un-clustered documents. In other words, an attempt to form new clusters among these database entities for which no appropriate existing cluster could be found.

5.2.4 Phase 4: Remapping the unique documents

At this point, certain number of semantically unique documents only left, for which no clusters nor other unique documents could be matched. Therefore, to visually show their *uniqueness*, these documents are reassigned to their first respective available **empty** BMU.

In case there is no available node, then, they should be mapped to the first BMU having un-clustered documents, in the consequence of which; will be formed new clusters of **un-clustered** documents UCl_i that could be colored differently. For example, the un-clustered documents and the clusters of un-clustered documents could be colored in blue to identify them as semantically unique, while the clusters of semantically similar documents could be colored in red to show that they are semantically similar. This way it would facilitate the user the localization of the similar or related entities that potentially could be mapped to together to form entity after integration

5.3 SOM's map update

The SOM map is updated with the redistribution of homogenous clusters of columns as well as the un-clustered ones. As an illustration, the map shown on Figure 16 is the map on Figure 17 (next page) after applying the CIBC proposed algorithm to it. In the case where there are no heterogeneous clusters UCl_k, then, any node having 2 entities or more would be considered as cluster Cl_k with semantically related data formed in the previous phases.

After updating the map, it can be clearly seen that the map is more balanced visually. This stability is a result of the enhancement of the clusters overall quality which is determined by the F-measure. More particularly, the clusters recall measure is significantly improved. More details can be found in section 6.4 or in [39].

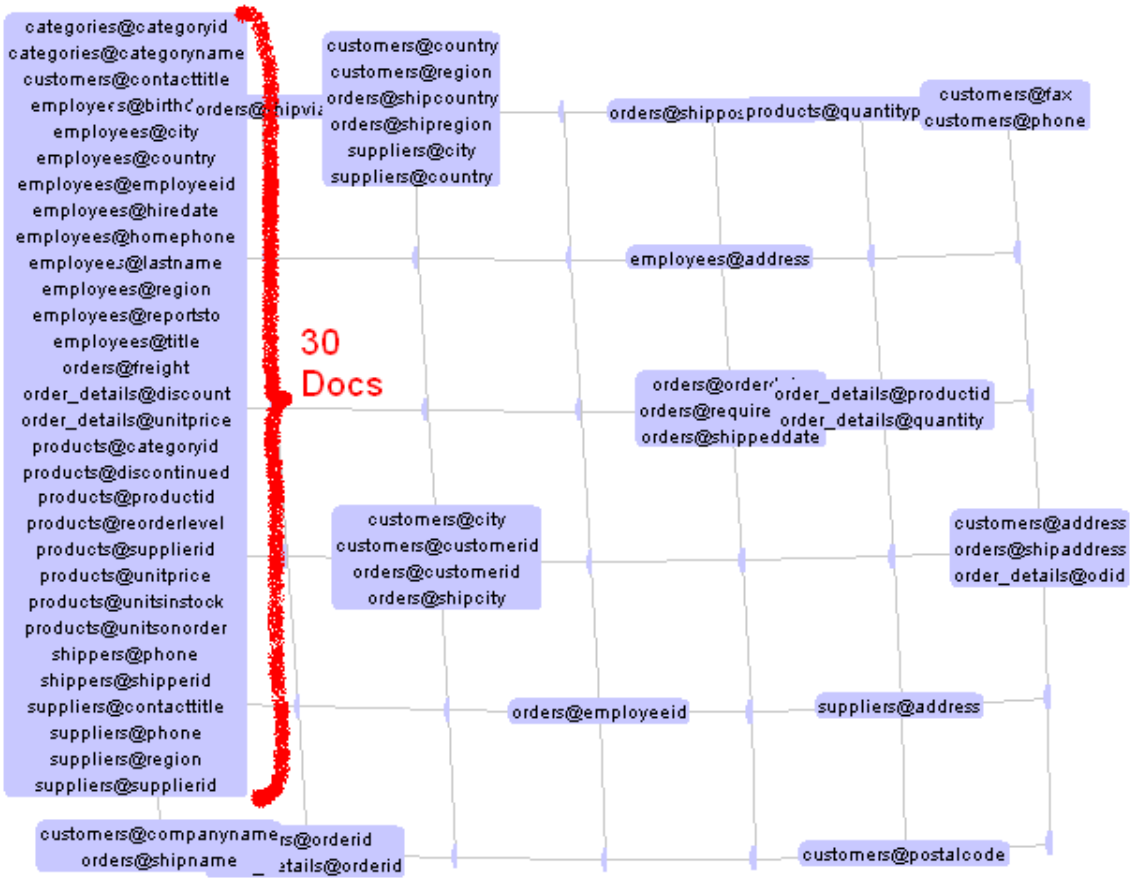


Figure 16 SOM's trained map

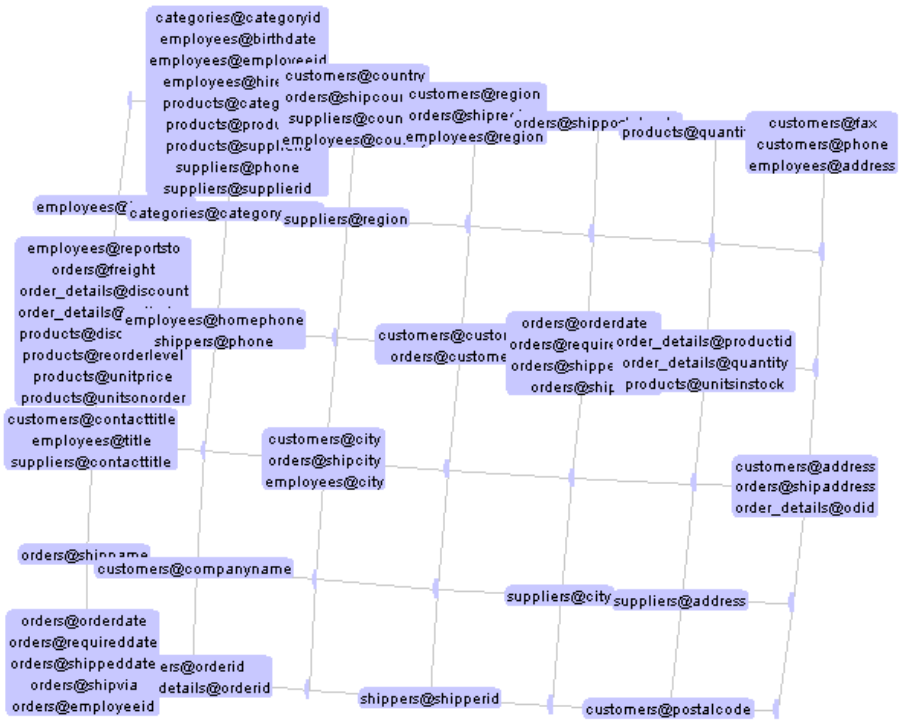


Figure 17 Updated SOM's map after CIBC post-processing

5.4 Post-processing overview

The proposed a new post-processing algorithm named common item-set based classifier complements adequately SOM by improving the clusters overall quality, particularly, it enhances significantly the clusters homogeneity (recall) formed by the SOM's trained map. CIBC can be applied to any data mining or machine learning algorithm, however it is more appropriate for algorithms lacking in clusters precision.

In addition, the development of CIBC permits to identify statistically insignificant terms such as stop words and avoid processing them. Thus, its data cleaning property could be used for pre-processing as well. In the future, if time permits, we aim to tune the CIBC algorithm, and test it with the remaining proposed techniques on other machine learning algorithms.

Chapter 6
Case Study

In the research project, experiments were conducted in two phases. The first test set served as prove of concept of the heterogeneous data classification on small scale, while the second was more experimental with a larger data repository.

As mentioned earlier, the first test set served to prove that it is possible to extract coincident meaning from heterogeneous data types [39]. The experiments served to try different pre-processing techniques and evaluate their impact on the results. For example, the selected database was small enough to permit the comparison of data processing with dimensionality reductions versus data processing without dimensionality reduction. In addition, the proposed post-processing method is evaluated and used to enhance the classification results.

The second test bands, serve to sharpen the results because of lesson learned from the first series of test. It was possible just by focusing on the pre-processing phase to enhance the classification outcomes significantly and meet the expected results.

In this chapter, both tests are presented as well as the results, followed by a conclusion from both case studies.

6.1 Corpus

The different proposed algorithms were tested on two *demo* databases available online: Northwind [50] and Sakila [53]. As shown in Table 6 databases used for case studies, the first database is smaller which permits to test the algorithms without dimensionality reduction. The second database is larger; hence, more representative of a real industrial databases.

Data set	Columns	All terms	Textual terms	Numeric Terms	Classes
Northwind	77	4681	2154	2527	15
Sakila	89	22104	4932	17172	20

Table 6 databases used for case studies

It is important to mention that the classes refer to the expected clustering results, while clusters refer to the algorithm actual clustering results. The latest ones are always compared to the classes for the evaluation of the algorithms performances. Note that the classes setup was done manually by examining the content of every database entity, and grouping them based on similar content.

Both databases represent small companies with the usual entities that can be found in such databases such as employee related information, products, costumers, orders, addresses, *etc.*

6.2 Pre-processing setup

In both case studies, the processing techniques described in the Chapter 3 were used. However, every case study focused on specific experimental objectives which are described below. Also, it is important to underling some information related to pre-processing.

The process of breaking a text up into its constituent tokens is known as tokenization. Because no linguistic pre-treatment were completed (*i.e.* lemmatization, stemming or stop words elimination), the impact of tokenization has more impact on the results. It is not the purpose of this paper to evaluate the impact of tokenization, however it is important to mention some important facts. For example, when we use bag of words as method of representation, if the non alphanumeric characters such as "()-+;." are not eliminated the results could be affected. In the first test series, the F-measure drops when using the SOM algorithm drops even by 15% and for the hybrid SOM and CIBC algorithm by 30%. Therefore, all the **non alphanumeric** characters are eliminated during our experiments.

6.3 Evaluation measures

One of the most used performance evaluation of unsupervised classifiers in the IR literature, with respect to the known classes for each document, are *F-measure* and *Entropy* which are based on *Precision* and *Recall* measures.

Recall that the difference between clusters and classes is that clusters are the real clusters resulted from the data mining algorithms, while classes are the expected ideal clustering results. Consequently, clusters are compared to classes in order to calculate the data mining algorithms performance and quality.

- **Precision:** measures how homogeneous or relevant are the SOM map's clusters.

  P = Precision $(i, j) = N_{ij} / N_j$

  =number relevant items retrieved / number of items retrieved

  where, N_{ij} represents the number of true positives, and N_j is the number of members in the cluster j.

- **Recall:** measures how many of the documents that should be retrieved, were really retrieved.

  R = Recall$(i, j) = N_{ij} / N_i$

  = number relevant items retrieved / *nb relevant* collection items

  where, N_{ij} represents the number of true positives, and N_i is the number of elements belonging to the class *i*.

- **F-measure**: measure the overall cluster quality. It distinguishes the correct classification of labels within different classes. In essence, it assesses the effectiveness of the algorithm on a single class and the higher it is; the better is the clustering. It is defined as follows

  $$F(i) = 2PR / P + R$$

- **Entropy**: measures the cluster homogeneity

$$E(i) = - \sum P * \log P$$

6.4 Case study one: Northwind

6.4.1 Experimental objectives

The objective of these tests is to validate the proposed mining of heterogeneous data types (numerical and textual) by unified vectorization. These tests serve to prove that is possible to extract coincident meaning from heterogeneous data types by simultaneous processing using UV. At the same time, the performance of the proposed post-processing algorithm CIBC should be evaluated. One of the main test focus is to verify homogeneity level of the clusters after post processing because that is the main reason of having developed the CIBC algorithm. Another experimental interest is to select the best text representation, among those proposed in the Chapter 3 (tf-idf, binary, bag of words, n-gram) in order to find the optimal clustering result.

6.4.2 Preliminary tests: without dimension reduction

A good classification requires a good presentation [31]. However, the vast number of text representation possibilities presented earlier requires selecting the most significant ones to continue the tests further. In this sense firstly will be tested on *Northwind* database without dimensionality reduction, different combination of tokenization and vectorization. Then, the best representations will be kept to continue further the tests with dimension reduction.

Recall that the main advantage of having chosen *Northwind* is that its size is relatively small to do tests without dimensionality reduction, and the semantic content is large enough to permit the evaluation of the clustering properties of the proposed techniques.

Two tokenization methods were used: bag of words versus N-gram described earlier in section 3.1.2. Besides, three vectorization techniques were selected (see section 3.1): binary for textual data, TF-IDF for textual data, histogram for numerical data. Additionally, TF-IDF was tested on textual and *numerical* data at the same time where the numbers were processed exactly as text.

		Data types					
		Text	Numeric	Text	Numeric	Text	Numeric
Processing	**Vectrz.** / **Tokeniz.**	Tf-idf	Histog.	Binary	Histo.	Tf-idf	Tf-idf
SOM	Bag words	58.24		49.85		54.37	
SOM+CIBC	Bag words	**87.64**		74.18		65.07	
SOM	3-Gram	51.55		45.12		51.26	
SOM+ CIBC	3-Gram	59.75		52.34		60.18	

Table 7 F-measure (preliminary tests)

The preliminary tests (Table 7 and Table 8) show an evident performance advance of the hybrid algorithm (SOM + CIBC) over pure SOM. There is an improvement of the quality of clustering precision of [7.22-29.4] % which is represented by the F-Measure. The entropy measure decreases by [7.93-20.39] % when applying SOM + CIBC techniques, which reflects the level of improvement of the homogeneity of clusters. The best results of the hybrid algorithm are when "bag of words" is used as a tokenizer, combined with the proposed TF-IDF and histogram as vectorization methods. However, surprisingly N-gram (3-gram) tokenization did not improve the results when used with proposed unified vectorization.

		Data types					
		Text	Numeric	Text	Numeric	Text	Numeric
Processing	**Vectrz. / Tokenz.**	TF-IDF	Histogram	Binary	Histogram	TF-IDF	TF-IDF
SOM	Bag words	23.72		26.90		23.65	
SOM + CIBC	Bag words	**8.44**		14.57		**3.26**	
SOM	3-Gram	28.33		27.83		27.08	
SOM + CIBC	3-Gram	14.76		19.90		**10.88**	

Table 8 Entropy measure (preliminary tests)

6.4.3 Test series one: with dimension reduction

Dimensionality reduction was applied for the term-document matrix of the Northwind database. The size of textual data vocabulary for all vectorization type was originally 2154 terms, then the dimension was reduced to 1000 using RP. However for the numerical data, RP was not applied because the dimension was already enough reduced using histograms.

From the results, we can see that the best performance in general is the usage of the proposed hybrid algorithm of SOM and CIBC. First of all, CIBC improves the overall SOM classification quality of by [4.84 - 23.78] % (F-Measure). Secondly, we can observe that the proposed heterogeneous data mining techniques for numerical and textual data work very well, particularly when “bags of words” is used as tokenization method for textual data. Finally, the most important remark is when we use the proposed hybrid algorithm (SOM and CIBC) with the suggested weighting measures (TF-IDF with

histogram); the quality of clusters homogeneity is impressive, as computed in Table 9. Sometimes even faultlessly as shown on Table 10. In brief, the clusters homogeneity is improved by [17.49-30.25] % according to the entropy measure.

	SOM			**SOM + CIBC**		
	Text	Numeric	Text	Text	Numeric	Text
Vectrz. / **Tokeniz.**	tf-idf	Histogram	tf-idf	tf-idf	Histogram	tf-idf
Bag of words	54.03		52.93	**71.42**		59.21
Bag of words	51.63		43.83	66.66		56.47
3-Gram	42.88		62.02	66.66		68.28
3-Gram	46.00		54.04	61.00		62.88

Table 9 F-Score values with dimension reduction

	SOM			**SOM + CIBC**		
	Text	Numeric	Text	Text	Numeric	Text
Vectrz. / **Tokeniz.**	tf-idf	Histogram	tf-idf	tf-idf	Histogram	tf-idf
Bag of words	22.44		27.09	**1.05**		5.86
Bag of words	27.26		32.34	0.00		9.77
3-Gram	30.25		21.55	0.00		4.06
3-Gram	28.59		25.55	0.90		2.85

Table 10 Entropy values with dimension reduction

The proposed a new algorithm (CIBC) complements adequately SOM by improving the clusters quality. More precisely, CIBC enhances significantly the homogeneity of SOM

clusters. In other words, CIBC offers good *recall* results while SOM has them very low. Therefore the combination of the two algorithm results in a more efficient overall clustering quality.

However, an interesting remark is that the proposed unified vectorization technique performs better when using "bag of words" tokenization for textual data rather than N-Gram. This is valid for all processing scenarios in these tests. Usually, N-Gram performs better for the traditional homogenous data type processing. Therefore, the next series of experiments will concentrate on how to optimize the clustering results by focusing on pre-processing. The results should be higher when using N-GRAM as vectorization method when processing heterogeneous textual and numerical data because it is usually higher for *pure* textual data mining.

6.5 Use case two: Sakila

6.5.1 Experimental objectives

In the previous tests, the results were not optimal because when N-Gram tokenization was used for heterogeneous data processing, the performance was lower. In this test series, the objective is to test a new measure named BF-IDBF described in Chapter 3 that should improve the results because it offers a similar representation to TF-IDF measure.

These tests serve to estimate the added value of the BF-IDBF on processing heterogeneous data types, and more specifically using SOM classification method. In order to measure the contribution of the BF-IDBF weight to the processing phase, F-measure is used. It is used with respect to the known classes for each document, and it is based on Precision and Recall weights.

It is important to mention that the Sakila database was chosen because it is closer to a real industrial database. Around 60 % of the files (database entities) are constituted of purely numerical data such as keys, dates, prices, phone numbers, *etc*. The remaining ones are textual or a combination of the two data types. Also, with around 17.000 terms *Sakila* corpus is much larger than *Northwind* database as described in the section 6.1.

6.5.2 Experiments

For every measure, four set of tests were completed as shown on Table 11, Table 12, and Table 13 (next page). Firstly, the SOM based classification of the database entities is evaluated without unified vectorization, *i.e.*, either numerical or textual exclusive input data was processed. In this case, the dimension is reduced using RP to 1500. Then, the heterogeneous textual and numerical data types are processed by unified vectorization using different vectorization including BF-IDBF. Therefore, it is possible to estimate the enhancement caused by the proposed measure. Note that in the second case, the dimension was reduced to 1250 for textual data and 750 for numerical data for an overall dimension size of 2000. In other words, there is a more important loss of information when the data is processed by unified vectorization, but we do not think it has a major impact to bias the results.

	Text only	**Text & numeric**	**Textual & numeric**	**Numeric only**
	TFIDF	**TFIDF+HISTO**	**TFIDF+BFIDBF**	**BFIDBF**
Bag words	26.68	46.23	64.81	-
3-Gram	21.68	38.15	58.77	-
4-Gram	30.02	42.72	**65.56**	-
5-Gram	26.57	47.28	63.94	-
Numeric	-	-	-	54.49

Table 11 Precision measure with different representations

	Text only	**Text & numeric**	**Textual & numeric**	**Numeric only**
	TFIDF	**TFIDF+HISTO**	**TFIDF+BFIDBF**	**BFIDBF**
Bag words	89.89	74.15	86.52	-
3-Gram	**93.26**	70.79	86.52	-
4-Gram	92.13	71.91	88.76	-
5-Gram	92.13	71.91	83.14	-
Numeric	-	-	-	69.66

Table 12 Recall measure with different representations

	Text only	**Text & numeric**	**Textual & numeric**	**Numeric only**
	TFIDF	**TFIDF+HISTO**	**TFIDF+BFIDBF**	**BFIDBF**

Bag words	41.15	56.95	74.11	-
3-Gram	35.18	49.58	69.99	-
4-Gram	45.29	53.59	**75.42**	-
5-Gram	41.25	57.05	72.29	-
Numeric	-	-	-	61.15

Table 13 F-measure with different representations

The experiments (Figure 18) show that the proposed combination of TF-IDF and BF-IDBF vectorization enhances the precision of SOM significantly by at least 15%. More specifically, the best results are obtained when using 4-gram as tokenizer which is an improvement by almost 20% of the SOM's precision. Surprisingly, the precision results obtained using exclusively BF-IDBF (54.49%) were better that even the combination of TF-IDF and histogram. That can be explained by the fact that portion of numerical data in sakila database is more important than textual data.

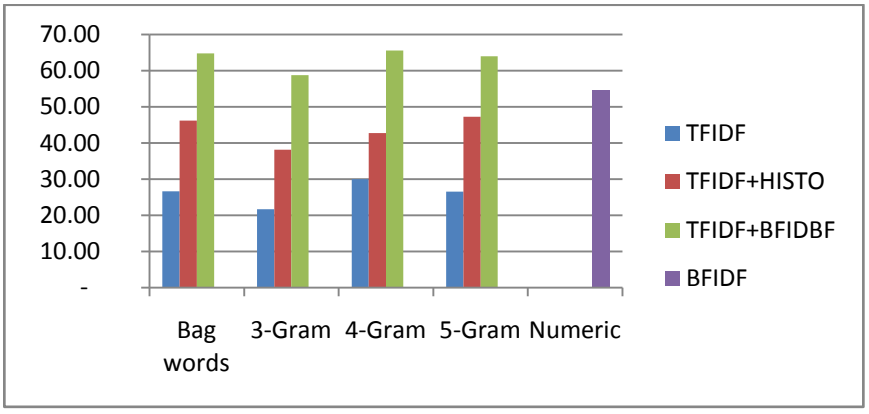


Figure 18 Precision measures with different representations

In regards to the Recall measure, the best performance observed was with the exclusive usage of TF-IDF vectorization, however, its precision was very low and that is why F-measure is a more objective way of comparing these representations. Then, the proposed combination of TF-IDF and the new BF-IDBF vectorization measures follow in the second position. It is interesting to note (Figure 19) that the usage of the exclusive BF-

IDBF measure with purely numerical data (69.66%) is almost as good as the unified vectorization by TF-IDF and histogram. This shows how the pre-processing phase is important in the whole data mining process; perhaps the most important. In this case, a more efficient pre-processing enhances the results. In addition, the processing time is reduced because the post-processing phase is eliminated.

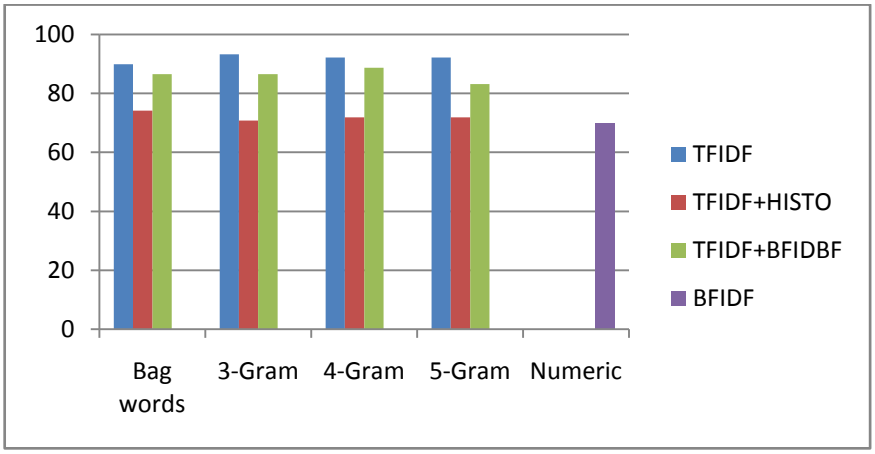


Figure 19 Recall with different representations

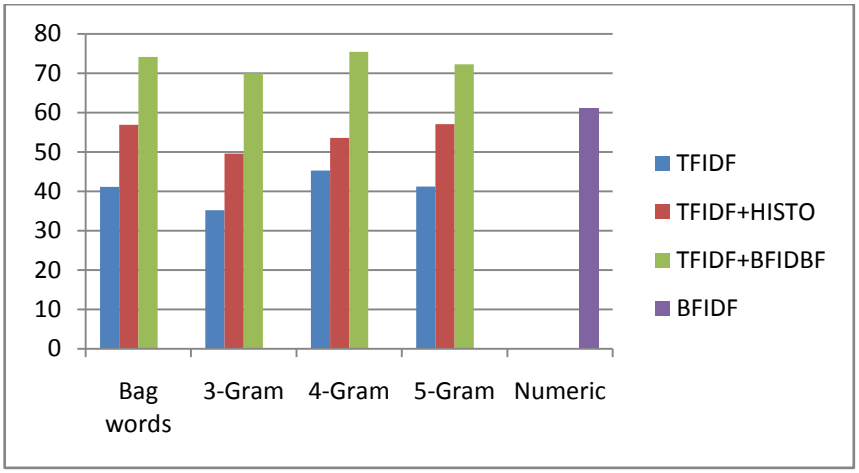


Figure 20 F-Measure with different representations

Finally, the precision and the recall are combined equally to produce the F-measure (Figure 20) which is more objective to compare the different representations. It can be easily observed, that 4-gram tokenization combined with the proposed vectorization using TF-IDF and BF-IDBF overcomes all the other representations. In fact, it performs better

than the unified vectorization by TF-IDF and histogram by around 20% and almost *doubles* the performance of the traditional pure textual processing using TF-IDF. Even the usage of the pure BF-IDBF representation of the exclusively numeric data performs better (61.15 %) than the pure TF-IDF or even the combination of TF-IDF and histogram. This demonstrates the beneficial properties of the proposed BF-IDBF measure. It shows as well that the pre-processing step is one of the most important phases in data classification because of the major impact on the machine learning algorithms result. In addition, per induction the proposed combination of TF-IDF and the new BF-IDBF measure can be applied to any other algorithm for probably better results.

6.6 Summary of experiments

Through the two presented case studies, several pre-processing methods were evaluated for heterogeneous data processing using SOM. In addition, a post-processing technique was introduced for better performances.

To begin with, the efficiency of simultaneous processing of heterogeneous data by unified vectorization was demonstrated. However, some unexpected effects were observed when the TF-IDF vectorization measure for textual data was combined with histograms vectorization measure for numerical data portion. Probably, these outcomes are the consequence of heterogeneity of the vectorization measures because these phenomena were not observed when the TF-IDF and BF-IDBF; hence, similar representation were used.

We can deduct two things from the previous results. Firstly, it is not necessary to have the same representations of the heterogeneous data in order to extract coincident meaning by unified vectorization mining. However, the results are better when similar representations are used for the heterogeneous data processing. In other words, if in the future other data types such as multimedia data types are to be processed by unified vectorization, it is not necessary to have the same representation of the data as the textual and numerical data for example. However, it is assumed that the results will not be as good as if the representation were similar. Furthermore, if the heterogeneous respective vectorized data

do not represent the same information, it is recommended to use post-processing techniques, such as CIBC, because it appeared to be efficient in such scenarios.

Chapter 7
Conclusion

In this thesis, we presented a data integration tool of unfamiliar heterogeneous textual and numerical data repositories. The automatic unsupervised classification and visualization were completed through the usage of SOMs. We demonstrated that by using a statistical data integration and visualization tool that it greatly facilitates data integration operations by showing semantic similarities among database entities that should be joined. This tool is applicable to data integration over web data sources, and tuples classification based on the semantic content.

A new approach for mining *simultaneously* heterogeneous textual and numerical data types by unified vectorization was proposed, which led for better coincident clustering and classification results. Additionally, a new weighting measure Bin Frequency – Inverse Document Bin Frequency (BF-IDBF) was for numerical data type was proposed; hence, it improved the precision significantly as well as the recall of the SOM algorithm. It is highly likely that this measure would improve the performance of any other machine learning algorithm for heterogeneous textual and numerical data processing. Finally, a post-processing algorithm Common Item-set Based Classifier (CIBC) was introduced to improve the homogeneity of the clusters (recall), which can be used with other data mining or machine learning techniques.

Some of our future work will focus on the improvement of CIBC, integrating other data types, applying the proposed pre-processing and post-processing techniques to other data types such as multimedia data types and metadata. Furthermore, we would like to test the unified vectorization by TF-IDF and BF-IDBF with other machine learning algorithms. Finally, we aim to apply the proposed techniques in other applications such as network intrusion detection, medical field, business intelligence, *etc*.

Bibliography

1. Eklund, T., Back, B., Vanharanta, H., Visa, A. *Benchmarking International Pulp and Paper Companies Using Self-Organizing Maps.* Turku, Finland : TUCS Technical Report No 396, Turku Centre for Computer Science, 2001.

2. Kloptchenko, A., Eklund, T.,Karlsson, J., Back, B.,Vanharanta,H., Visa, A. *Combining data and text mining techniques for analysing financial reports.* Intelligent Systems in Accounting Finance & Management, Vol. 12, no 1, 2004. pp. 29 - 41.

3. Back, B., Toivonen, J., Vanharanta, H., Visa, A. *Comparing numerical data and text information from annual reports using self-organizing maps.* International Journal of Accounting Information Systems, Vol. 2, no 4, 2001. pp. 249-269.

4. Rábová, I., Konečný, V., Matiášová, A. *Decision Making with Support of Artificial Intelligence.* Agricultural Economics, Vol. 51, no 9, 2005. pp. 385–388.

5. Parvizian, J., Tarkesh, H., Farid, S., Atighehchian, A. *Project Management Using Self-Organizing Maps.* Industrial Engineering and Management Systems, the official journal of APIEMS, Vol.5, no 1, 2006.

6. Hearst, M. A. *Untangling Text Data Mining.* Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, College Park, Maryland, USA, 1999. pp. 3-10.

7. Pyle, D. *Data Preparation for Data Mining.* San Francisco : Morgan Kaufman Publishers, 1999.

8. Lenzerini, M. *Data Integration: A Theoretical Perspective.* PODS'02, 2002. pp. 233-246.

9. Miller, R., Haas, L. M., A. Hernandez, M. *Schema Mapping as Query Discovery.* VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases, San Francisco, CA, USA, 2000. pp. 77-88.

10. Rahm, E. and Bernsteinp, P. A. *A survey of approaches to automatic schema matching.* The VLDB Journal, Vol. 10 , no 4, 2001. pp. 334-350.

11. Noy, N. F. *Semantic Integration: A Survey Of Ontology-Based Approaches.* SIGMOD Record, Vol. 33, no 4, 2004.

12. Wache, H., Veogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hubner, S. *Ontology-Based Integration of Information - A Survey of Existing Approaches.* The Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI), Victoria, BC, Canada, 2001. pp. 108-117.

13. Shvaiko, P. and Euzenat, J. *A Survey of Schema-Based Matching Approaches.* Journal on Data Semantics, Vol. 4, 2007. pp. 146-171.

14. Wikipidia. Data Integration. [Online] 2001. [Cited: 08 05, 2009.] http://en.wikipedia.org/wiki/Data_Integration.

15. BEA. BEA WebLogic Workshop. [Online] 1994. [Cited: 08 04, 2009.] http://www.bea.com/framework.jsp?CNT=demos.htm&FP=/content/products/workshop/learn.

16. Lau, C. *Developing XML web services with websphere.* IBM Systems Journal, Vol. 41, no. 2, 2002.

17. TIBCO Solutions (SOA) Resource Center. [Online] 2000. [Cited: 08 04, 2009.] http://www.tibco.com/solutions/soa/default.jsp.

18. XML Mapping Tools from Altova. [Online] 2005. [Cited: 08 04, 2009.] http://www.altova.com/solutions/xml-mapping-tools.html.

19. Stylus Studio XSLT Mapper. [Online] 2005. [Cited: 08 04, 2009.] http://www.stylusstudio.com/xslt_mapper.html.

20. XSLT. Cape Clear XSLT Mapper. [Online] http://www.capescience.com/education/tutorials/index.shtml#cc5.

21. Sonic Software Integration Workbench. [Online] 2004. [Cited: 08 04, 2009.] http://www.sonicsoftware.com/products/docs/integration_workbench_0604.pdf.

22. ASPN. XSLT. *ActiveState Visual XSLT.* [Online] 2006. [Cited: 08 04, 2009.] http://aspn.activestate.com/ASPN/XSLT.

23. Robertson, G. G., Czerwinski, M. P., Churchill, J. E. *Visualization of Mappings Between Schemas.* ACM SIGCHI Conference on Human Factors in Computing Systems, Portland, Oregon, USA, 2005. pp. 431-439.

24. Colomb, R. M. *Impact of Semantic Heterogeneity on Federating Databases.* The Computer Journal, Vol. 40, no 5, 1997. pp. 235-244.

25. Sheth, A. P. *Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics.* Norwell, Massachussetts, USA : M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.) Kluwer, Academic Publishers, 1998. pp. 5-30.

26. Salton, G. *Automatic Text Processing.* MA : Addison-Wesley, 1989.

27. Van Rijsbergen, C.J. *Information Retrieval 2nd ed.* s.l. : Butterworth-Heinemann, 1979.

28. Han, J., Kamber, M. *Data Mining, Second Edition: Concepts and Techniques.* San Francisco : Morgan Kaufmann, 2006, pp. 72-97.

29. R. Baeza-Yates and R. Ribeiro-Neto, eds. *Modern Information Retrieval.* s.l. : Addison Wesley Longman, 1999.

30. Lagus, K. *Text Mining with the WEBSOM.* PhD thesis, Department of Computer Science and Engineering, Helsinki University of Technology, 2000.

31. Amine, A., Elberrichi, Z., Bellatreche, L., Si- Monet, M., Malki, M. Concept-based clustering of textual documents using SOM. *In Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, Doha, Quatar.* 2008.

32. Aas, K., Eikvil, L.,. *Text categorization: a survey.* Oslo, Norway : Norwegian Computing Center, 1999.

33. Sahami, M. *Using Machine Learning to Improve Information Access.* PhD thesis, Computer Science Department, Stanford University, 1999.

34. De Loupy, C. *L'apport de connaissances linguistiques en recherche documentaire.* 16ème conference sur le Traitement Automatique des Langues Naturelles (TALN01), Tours, France, 2001.

35. Miller, E., Shen, D., Liu, J., Nicholas, C. *Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System.* Journal of Digital Information, Vol. 1, no 5, 2000.

36. Y. Miao, V. Keelj, and E. Milios. *Document Clustering Using Character N-Grams:A Comparative Evaluation With Term-Based and Word-Based Clustering.* 14th ACM International Conference on Information and Knowledge Management, Bremen, Germany, 2005.

37. Sebastiani, F. *Machine learning in automated text categorization.* ACM Computing Surveys, Vol. 34, no 1, 2002. pp. 1-47.

38. Eklund, T., Back, B. , Vanharanta, H., Visa, A. *Assessing the feasibility of Self-Organizing Maps for data mining financial information.* 10th European Conference on Information Systems (ECIS), Gdansk, Poland, 2002. pp. 528-537.

39. Bourennani, F., Pu, K. Q., Zhu., Y. *Visualization and Integration of Databases using Self Organizing Maps.* International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA'09), IEEE computer society and CPS, Cancun, Mexico, 2009. pp. 155-160.

40. Bourennani, F., Pu, K. Q., Zhu., Y. *Visual Integration Tool for Heterogeneous Data Type by Unified Vectorization.* Proceedings of the 10th IEEE International Conference in Reuse and Integration (IRI'09), LAS-VEGAS, USA, 2009.

41. Wang, J. *Data mining: opportunities and challenges.* USA : Hershey, 2003. pp. 323-349.

42. Fradkin, D., Madigan, D. *Experiments with Random Projections for Machine Learning.* Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, D.C, USA, 2003. pp. 517 - 522.

43. Bingham, E. and Mannila, H. *Random projection in dimensionality reduction: Applications to image and text data.* Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA, 2001. pp. 245 - 250.

44. Song, M., Wu, YF (eds.). *Handbook of Research on Text and Web Mining Technologies.* USA : Idea Group Inc., 2008.

45. K. Lagus, S. Kaski, and T. Kohonen. *Mining massive document collections by the WEBSOM method.* Information Sciences, Vol.163, 2004. pp. 135-156.

46. Kohonen, T. *Self-organizing formation of topologically correct.* Biological Cybernetics, Vol.43, no 1, 1982. pp. 59-69.

47. Kohonen, T. *Self-Organizing Maps.* Berlin : Springer-Verlag, 2001.

48. Zhang, J. *Visualization for Information Retrieval .* Berlin, Heidelberg : Springer-Verlag, 2008.

49. Burkhard, R. A. *Learning from Architects: The Difference between Knowledge Visualization and Information Visualization.* Ninth International Conference on Information Visualisation, London, UK, IEEE Computer Society, 2005.

50. Microsoft. Northwind. [Online] 2005. [Cited: 06 28, 2009.] http://www.microsoft.com/Downloads/details.aspx?FamilyID=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en.

51. Lin, X. *Map displays for information retrieval.* Journal of the American Society for information Science, Vol. 48, no 1, 1997. pp. 40-54.

52. Chang, S. J. & Rice, R., E. Browsing: A multidimensional framework. *Annual Review of Information Science and Technology, Vol. 28.* 1993, pp. 231-276.

53. MySQL. *Sakila.* [Online] 2005. [Cited: 06 28, 2009.] http://dev.mysql.com/doc/sakila/en/sakila.html.

Image References

Figure 5 http://www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml

Figure 6 http://www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml

Figure 7 http://www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml

Figure 8 K. Lagus, S. Kaski, and T. Kohonen**.** *Mining massive document collections by the WEBSOM method.* Information Sciences, Vol.163, 2004. pp. 135-156.

Figure 9 Robertson, G. G., Czerwinski, M. P., Churchill, J. E., Visualization of Mappings Between Schemas, ACM SIGCHI, 2005,Portland, Oregon, USA