

Automatic Text Ontological Representation and Classification via Fundamental to Specific Conceptual Elements (TOR-FUSE)

by
Amir Hossein Razavi

Thesis Submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
For the PhD degree in Computer Science

The Ottawa-Carleton Institute for Computer Science
Faculty of Engineering
University of Ottawa

©Amir Hossein Razavi
Ottawa, Canada,
2012

Abstract

In this dissertation, we introduce a novel text representation method mainly used for text classification purpose. The presented representation method is initially based on a variety of closeness relationships between pairs of words in text passages within the entire corpus. This representation is then used as the basis for our multi-level lightweight ontological representation method (TOR-FUSE), in which documents are represented based on their contexts and the goal of the learning task. The method is unlike the traditional representation methods, in which all the documents are represented solely based on the constituent words of the documents, and are totally isolated from the goal that they are represented for.

We believe choosing the correct granularity of representation features is an important aspect of text classification. Interpreting data in a more general dimensional space, with fewer dimensions, can convey more discriminative knowledge and decrease the level of learning perplexity. The multi-level model allows data interpretation in a more conceptual space, rather than only containing scattered words occurring in texts. It aims to perform the extraction of the knowledge tailored for the classification task by *automatic creation of a lightweight ontological hierarchy of representations*.

In the last step, we will train a tailored ensemble learner over a stack of representations at different conceptual granularities. The final result is a mapping and a weighting of the targeted concept of the original learning task, over a stack of representations and granular conceptual elements of its different levels (hierarchical mapping instead of linear mapping over a vector).

Finally the entire algorithm is applied to a variety of general text classification tasks, and the performance is evaluated in comparison with well-known algorithms.

Acknowledgements

First of all, I would like to thank the omnipresent God, who answers my prayers; the Lord who give me the strength to plod through the life harshness; thank you so much Dear God.

I never forget the eminent role of my late mum in any success I achieve and also her insights she has shared with me. A person who always generously encouraged me to wade through the future; whom, sadly, I can never kiss her hands again.

It would not have been possible to write this doctoral dissertation without the support, help and guidance of the kind people around me who in one way or another contributed and extended their valuable assistance; the people to only some of whom it is possible to give particular mention here.

My angle wife (Dr. Nassim Nassimi), for her kindness, patience and untiring effort at all our shared life. A lovely wife who has done every possible support throughout, for which my mere expression of thanks is never suffice.

My father (Dr. Mahdi Razavi) who has been my inspiration as I hurdle all the obstacles since my childhood and especially for his moral support during my PhD program.

My wife parents, for all their kindness, patience and steadfast encouragement to complete the degree; despite the distance, they have given me their unequivocal support.

The utmost gratitude to my supervisor, Prof. Stan Matwin, a great man, whom there is no need to mention about his unsurpassed knowledge and insight. He always granted me unlimited support, advice and friendship, not only by words also by his great heart, for which I am extremely grateful.

I would like to have special thanks to Prof. Diana Inkpen for her kindness, support and shared valuable insights in the relevance of the completion of this dissertation.

Prof. James R. Green, somebody without his helpful hands and kindly heart, I would have quitted my education many years ago.

Amongst my fellow graduate students in the Department of Computer Science of University of Ottawa, whom I personally grateful them all, the efforts, help and sympathy of Dr. Jelber Sayyad Shirabad, Dr. Md. Anisur Rahman and Dr Fazel Keshtkar stand as good examples.

I would like to acknowledge the financial, academic and technical supports of the University of Ottawa especially at the school of information technology that provided me an appropriate condition during my PhD program.

Last, but by no means least, I thank my cousin Mr. Ali Razavi and my friends Mr. Reza Tashvighi and Mr. Reza Meghrazi here and elsewhere for their support and encouragement.

Table of Contents:

Abstract	ii
Acknowledgements.....	iv
Table of Contents:.....	vi
List of Figures	viii
List of Tables	x
Glossary	xii
1. Introduction	1
1.1. Content and Motivation	1
1.2. Hypothesis.....	3
2. Background	5
2.1. Low-level Text Representation	5
2.2. Dimensionality Reduction.....	11
2.3. Definitions of Ontology in Knowledge Representation	18
2.4. Generality and Granularity of Knowledge	22
2.5. Definition of the Light-weight Ontology	24
2.6. Clustering	26
2.6.1. Distances in High Dimensional Data	27
2.6.2. Hierarchical clustering.....	32
2.6.3. Partitional clustering	34
2.6.4. Fuzzy c-means clustering.....	34
2.6.5. Spectral clustering (Kernel principal component analysis)	35
2.6.6. Density-based clustering.....	36
2.6.7. Subspace Clustering.....	36
2.6.8. Projected Clustering.....	38
2.6.9. Hybrid Approaches	39
2.6.10. Text Clustering.....	39
2.7. Ensemble learning.....	40
2.7.1. Bagging	43
2.7.2. Boosting	45
2.7.3. Voting	46
2.7.4. Ranking	48
2.7.5. incentive methods	48
3. Second Order Soft Co-Occurrence (SOSCO) Text representation	50
3.1. Preprocessing	51
3.2. Closeness Matrix	51
3.3. Soft Co-Occurrence Matrix	52
3.4. General and Domain Specific Stop Words Removal	57
3.5. Text Representation through Representation of the Containing Sentences	59
3.5.1. Sentence Second Order Representation Vectors.....	59
3.5.2. Document Representation Vectors	60
3.6. Contrast Parameter	64
3.7. An Example.....	67
3.8. Advantages of the SOSCO approach	70
3.9. Limitations	74

4.	SOSCO in Practice	76
4.1.	Classification of Emotional Tone of Dreams	76
4.1.1.	Dream Analysis in Psychology	79
4.1.2.	Dream Bank	80
4.1.3.	Sample Dream.....	80
4.1.4.	Methodology	81
4.1.5.	Classification Results.....	86
4.1.6.	Subjective Emotion Estimation Based on the other Emotions	89
4.1.7.	Discussion	90
4.2.	Classifying Biomedical Abstracts Using Collective Ranking Technique.....	92
5.	Automatic Text Ontology Representations	97
5.1.	Overview	97
5.2.	Conceptual Representation versus Low-level Feature Representation	98
5.3.	Extraction of the Conceptual Elements (Candidate Nodes of Ontology) via Clustering.....	100
5.4.	Fundamental to Specific Extraction of the Conceptual Elements	103
5.5.	TOR-FUSE Complete Algorithm and Flow-chart	104
5.6.	Text Ontological Representations via Hierarchical Clustering of Features	107
5.7.	Stack of Fundamental to Specific Representations	108
5.8.	TOR-FUSE – Classification process (Ensemble learning)	109
5.8.1.	Learning over the Fundamental to Specific Ontological layers.....	110
6.	Experiments and Results	114
6.1.	Reuters Transcribed Subset.....	114
6.2.	ISEAR Dataset	118
6.3.	The AMAN dataset	121
7.	Discussion	126
7.1.	TOR-FUSE vs. LSI	126
7.2.	partial Text Progression Representation (pTPR) - a motivation for future work	127
7.3.	A Few Notes on TOR-FUSE.....	129
7.4.	TOR-FUSE Limitations	130
7.5.	Contributions.....	131
7.6.	Conclusion and Future Works.....	132
	Appendix.....	135
	Multi-level Text Representation and Classification	135
	Offensive Language Detection	135
	Preliminary Definition and Discussion.....	136
	Related Work	138
	Flame Annotated Data	140
	Methodology	142
	Insulting and Abusing Language Dictionary	142
	Multilevel Classification.....	144
	Results.....	145
	Discussion	149
	Conclusion	150
	Bibliography	151

List of Figures

Figure 1. Graphical model representation of the aspect model in asymmetric (a) and symmetric (b) parameterization.	17
Figure 2. An example of ontology [Usc04]. The dashed line indicates the approximate possible border between the Upper Ontology and Domain Ontology.	21
Figure 3. Types of Ontology [Usc04]. The diagonal line indicates the approximate border between formal ontology and lightweight ontology.	25
Figure 4. An Example of Subject/Topic clustering.	27
Figure 5. Spiking Hyper-cube [Hec91].....	28
Figure 6. Density-based clustering, encloses and separates clusters using a density parameter.....	36
Figure 7. Sample dataset with four clusters, each in two dimensions in which the third dimension is considered as noise. Points from two clusters can be very close together, but this may confuse many traditional clustering algorithms. [Par04]	37
Figure 8. Illustrates the configurations of word pairs which can be extracted from a sentence (ending by either of the [. ! ?] symbols).	53
Figure 9. Sentence Average Vectors (Second Order Representation) Input: Soft closeness vectors of all words in each sentence; Output: Sentence average vectors (Second Order Representation)	60
Figure 10. Document Representation Vector.....	61
Figure 11. This graph shows how the SOSCO representation of a sentence in the corpus contains about 3 quarters of m , the corpus feature space attributes with non-zero values; $V_{wi} = w_i$. The actual presentation of each line is a sequence of dots with some gaps in between, which for better illustration and simplicity has been depicted as a solid line. Axis X is the feature space (words) while the axis Y is about all the representation vectors that participate in the aggregation process.	62
Figure 12. This graph shows how the SOSCO representation of a document in the corpus contains about 90% of the corpus feature space attributes with non-zero values; $V_{si} = S_i$. The actual presentation of each line is a sequence of dots with some gaps in between, which for better illustration and simplicity has been depicted as a solid line. Axis X is the feature space (words) while the axis Y is about all the representation vectors that participate in the aggregation process.	62
Figure 13. The above figure illustrates the vector representation of the word “Heart” as an example over the entire domain of the task (corpus).	68
Figure 14. Example of illustrative vector representation of the other word “Africa” over the entire corpus.	69
Figure 15. Illustrative vector representation of the first sentence: “ <i>The Congo River (also known as the Zaire River) is the largest river in the heart of Africa.</i> ” over the entire corpus. Values of the underlined attributes (from G1, G2 and G3) are well tuned (i.e. are considerable if in G3, and diminished otherwise) after the aggregation process.	70
Figure 16. Word order Text Progression Representation (closeness based) graph. Plot shows the closeness between the vectors for the bigrams in position n and the vector for the position $n+1$ (for example) in a given document.	71
Figure 17. Affection Polarograph (onirogram): An illustration of the polarity and emotional tone of the contextualization of dreams over the time.	84

Figure 18. Each triangle illustrates a candidate node (conceptual element) for the ultimate lightweight ontological representation. A candidate ontological node is applied as a new feature for domain representation. The intersections between the ontological nodes (darker areas) depict the existing relationship/dependency between the features (conceptual elements).	97
Figure 19. Targeted concept may be modeled /extracted through the nested multi-level TOR-FUSE	109
Figure 20. The idea of Partial Text Progression Representation (pTPR) for three sample conceptual elements. Each conceptual element specifically may be considered as a concept which explains the current text over the entire domain. These conceptual elements, may explain the text in different levels of the stack of fundamental to specific layers of TOR-FUSE.....	129

List of Tables

Table 1. Sentence based Closeness Matrix (Symmetric) MC ; an illustration of how closely words are interrelated	56
Table 2. Closeness Word Vector	58
Table 3. An example portion of the modifier table in the system. Each value is added to the initial affect value extracted from LIWC, considering the initial affect sign and the above corresponding value sign.....	83
Table 4. An example of the applied affection modification. The first two columns are based only on the LIWC dictionary, and the last two columns show the weights after modifications.....	84
Table 5. The attributes extracted from the onirogram (directly or indirectly).....	86
Table 6. Attribute selection results	86
Table 7. Results of our best classifiers applied on each of the attribute subsets individually.	88
Table 8. Emotion estimation based on the other emotion rates (0-3).	90
Table 9. Confusion matrix on the prediction zones applying ensemble BOW and SOSCO by voting in a committee of classifiers.	94
Table 10. Performance Evaluation.....	95
Table 11. False Positives with respect to data representation methods. Numbers are out of the 8,700 articles in the process.	96
Table 12. False Negatives with respect to data representation methods.....	96
Table 13. Comparison of the chosen evaluation measures of the classifiers for different representation methods. BOW+LSA is the LSA transformation of the explained BOW features; SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE Representation method with the BOW representation as level 0 of its stack (a level lower SOSCO which is in the level one).	118
Table 14. Comparison of the classification evaluation measures for different representation methods.	121
Table 15. Comparison of the classification evaluation measures for different representation methods. BOW+LSA is the LSA transformation of the explained BOW features; SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE Representation method with the BOW representation as level 0 of its stack (a level lower SOSCO which is in the level one).	123
Table 16. Shows 20 paired t-tests that compare the correctly classified percentage (accuracy) of the four mentioned classifiers over the different representations. BOW+LSA which is the LSA transformation of the explained BOW features is also our tests' base; SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE representation method with BOW representation as level 0 of its stack; 'v' is the sign of victory or significant difference over the test base (the first column of each row).....	125
Table 17. Flattened confusion matrices for all 6 classification results. True Pos. shows the number of texts which correctly classified as Okay; False Pos. shows the number of texts which falsely classified as Okay; True Neg. shows the number of texts which correctly classified as Flame, and False Neg. shows the number of texts which falsely classified as Flame.....	146

Table 18. Performance comparison along the three levels of classifications, for cross-validation (C.V.) of the training data and the test set.	147
---	-----

Glossary

Term	Page
Closeness	52
Closeness Matrix	51
Conceptual Elements.....	19
Configuration of word pairs.....	53
Context	1
Contrast parameter.....	64
Co-Occurrence	8
Curse of dimensionality.....	28
Density-Based clustering.....	36
Domain ontology.....	22
Ensemble learning.....	40
First order text representation	8
Formal ontology	25
Fundamental to specific.....	24
Fuzzy.....	23
Generality	22
Granularity.....	22
Hierarchical Representation.....	32
Light-weight ontology	24
LSA/LSI.....	13
Mahalanobis distance function	30
partial Text Progression Representation (pTPR)	127
Second order text representation	8
Sentence vector representation.....	59
Smoothness.....	63
Soft Co-occurrence.....	50
Soft-clustering.....	107
SOSCO	61
Spectral clustering.....	35
Stack of representations.....	108
Stacking.....	40
Subspace clustering.....	36
Text Progression Representation (TPR).....	71
Text vector representation.....	59
TOR-FUSE algorithm	104
Upper ontology.....	21
Window size.....	6
Word vector representation.....	58

1. Introduction

1.1. Content and Motivation

According to knowledge representation and extraction surveys, classification of text data has been a practical and effective task for many years. In order to improve the performance of text mining and knowledge extraction, approaches require a quantitative method to represent contexts¹ [Pan03] that is as informative and expressive as possible. Hence, efforts to create a more accurate, informative and expressive representation can be considered essential steps toward the expressed goal. In this regard, if we consider the words as the smallest informative unit of a text, there is a variety of well-known quantitative information related to them which could be helpful in representing the text. For example, familiar methods include the appearance or non-appearance of a word in the corpus inside individual documents, the frequency of each word in the document, and some type of normalized frequency of the containing words of a document over the corpus feature space (e.g., tf-idf [Spa72]). In some applications, this kind of information is extracted from a limited number of consecutive words in a context, such as bi-grams, tri-grams, etc.

Such representation methods have been used in a variety of information extraction projects, and in many cases have even outperformed some syntactic-based methods. However, we should remember that these kinds of representations are information islands surrounding the words which are extracted from the text stream, and we have lost

¹ By “Context” we mean a part of a text or statement that surrounds a particular word or passage (topic) and somehow explains, expands or interprets its concept (The American Heritage. Dictionary of the English Language, Fourth Edition. Boston: Houghton Mifflin Company; (2000).)

valuable knowledge that could be inferred by considering the different types of relations between the words. These major relations are actually the essential components that, at a higher level, could express concepts or explain the main trend of a text. For example, Bergadano et al. [Ber92] introduced the idea of a two-tiered concept representation method, and explained how concepts in such a representation can be learned from examples. They believe that in a two-tiered representation, the first tier can capture basic concept properties, while the second attributes some concept modifications and context dependencies. The goal of their representation was explained as capturing and representing different degrees of typicality of instances.

Delin et. al. introduced genres of documents based on analysis at five levels: content structure, rhetorical structure, layout structure, navigation structure and linguistic structure [Del02a]. Miller defines genres as, “typified rhetorical actions” that respond to *recurring situations* and become instantiated in groups’ behaviors [Car84]. Genres also represent constellations of textual attributes; some attributes are necessary and other attributes are optional (Perry’s definition). [Per08]

Moreover, context plays an important role in shaping genres. “Genre theory does not conceptualize context as simply the space outside of text or the container surrounding texts, but as dynamic environments that simultaneously structure and are structured by the communicative practices.” [Eme04].

To summarize these theories, a representation method which could add different levels of relations and dependencies to the raw information items, and illustrate the characteristics of text in a more extensive manner, could play an important role in knowledge extraction, concept analysis and sentiment analysis tasks.

In this dissertation, the main focus is on how we represent our domain. Thus, we first introduce our new representation method, which is designed, implemented and supported by experimental results. In the second stage, we build a hierarchy of “fundamental to specific ontological² representation layers” based on the introduced representation. In the third stage, we tailor an ensemble learning method to exploit a composition of the hierarchical representation layers, in order to explore the most discriminative classification model for the task of contextual classification. Since the thesis consists of components of the three subjects of *text representation*, *ontology extraction via hierarchical clustering* and *ensemble learners*, some corresponding literature on selected scenarios will be discussed separately in the background chapter.

1.2. Hypothesis

The main intent of this thesis is to present a novel approach to high dimensional problems in Machine Learning (ML) in general, and in text analysis and knowledge representation in particular. Semantic representation and conceptual analysis could be improved if we apply a standard shared vocabulary to express the main content while coping with the polysemic issues. In such a space, we would likely observe that semantics is represented in a considerably less ambiguous form, and can more simply feed machine learner algorithms. Our simple approach for choosing the intended meaning among the underlying vocabulary senses is taking the “second order” relation of the vocabulary into account. In this approach, we amplify the learning performance by applying indirect (second order) similarity relations (closeness) of the words. Then, we

² Lightweight Ontology - see Section 2.5. for more details

use the second order representation method of the words, in order to analyze and represent its containing corpus accordingly. In such a space similarities among words and texts are also calculated based on their corresponding second order representation vectors.

We believe that integrating the second order representation of a corpus with the conventional BOW representation can boost the supervised learning performance on textual tasks. We show the veracity of that hypothesis empirically in Chapter 6.

2. Background

In this chapter we include related literature and background materials to this dissertation. In section 2.1 and 2.2 we review works related to text representation in general, which can be regarded as an introduction for chapter 3 and 4. Sections 2.3 to 2.6 consist of introductory material and literature for chapter 5 and 6. In the last part of this chapter, section 2.7 brings some material about ensemble learning in general as pre-requirement material for chapter 7.

2.1. Low-level Text Representation

To solve many types of problems by machine, we first need to represent extensive knowledge in a quantitative manner, and then apply a suitable type of AI algorithm. Text representation is one of the areas which require the development of an appropriate knowledge representation, prior to any kind of knowledge extraction algorithm.

We normally use terms/words as the smallest meaningful unit of any context, and they play a very important role in expressing any meaning or intention through text.

There are a variety of Vector Space Modeling (VSM) methods which have been well explained and compared in [Tur10]. In Natural Language Processing (NLP) in general, and supervised text mining in particular, the most common method for context representation is 'Bag-Of-Words' (BOW). With this method, text (i.e. a sentence or a document) is represented by the words it contains, and the word order—and, consequently, the grammar and context—are ignored. Probably the starting point for the idea of the BOW representation in a linguistic domain is found in Zellig Harris's 1954 article, *Distributional Structure* [Har54]; in that paper he conducted studies on the words

individually and not in the context. BOW representations vary by the way each word is represented, and it is most appropriate for dictionary-based modeling, in which each document is represented as a 'bag' (since the order is ignored) containing a number of words from the dictionary. If the focus is on the presence or absence of a word it is called a binary representation, whereas if the frequencies of the words are also taken into account it is called a frequency representation. A normalized frequency representation is the tf-idf (term frequency-inverse document frequency) [Spa72] method, which is appropriate for a variety of classification tasks. tf-idf is a good method for weighting the elements in the representation, where each element (each term's assigned value) is proportional to the number of times the term appears in each document. With this method, the weights of rare terms are increased to reflect their relative importance to the containing text.

For the BOW representation, additive smoothing could be helpful for assigning non-zero probabilities to words which do not occur in the sample. Chen & Goodman (1996) empirically compare additive smoothing to a variety of other similar techniques which can be considered as an introduction for a potentially applicable smoothing method on BOW [Che96].

Capturing the right sense of a word in its context is a critical issue in representation methods. When we review the literature in this area, there are several hypotheses:

- You shall know a word by the company it keeps [Fir57];
- We can do Latent Relation Mapping (LRM) between two set of words only based on their co-occurred words in a small window size [Tur08];

- Meanings of words are (largely) determined by their distributional patterns (Distributional Hypothesis [Har64]);

- Words that occur in similar contexts will have similar meanings [Mil91];

Most efforts at semantic extraction of words are focused on semantic similarity [Man98]: “Automatically acquiring a relative measure of how similar a word is to known words [...] is much easier than determining what the actual meaning is.” There are many works about semantic similarity based on the Distributional Hypothesis [Har85], [MCD02], which states that words which occur in similar contexts tend to be similar.

Therefore, similarity can be a good approach to the essential problem of similarities between words. For example, the categorical status of concepts is usually estimated by similarities [Ros78], [Smi81] or even calculating these similarities taking to account the word-word distances [Lun95], [Lun96], and these similarities can be calculated even based on some lexical databases such as WordNet [Pan05] or HowNet [Pen07]. Mihalcea et. al. present a method which uses corpus-based and knowledge-based similarity measures for semantic analysis over short text corpora [Mih06].

Similarity also helps with word sense disambiguation, which is normally done according to the level of similarities of different senses of the targeted word in the context. In addition to being considered independent primitives with the same role as first order text representations [Pos68], [Ros78], similarities can also be taken into account dependently over common attributes (features), like second order co-occurrence

³[Sch95], [Tur11]⁴ ; however the mentioned second order co-occurrence is not directly related to the concept of “second-order logic”.

In unsupervised concept learning and word sense disambiguation, there are different approaches to represent a given target word⁵, or a context (which has been defined at the first footnote) with no target word; including first order word co-occurrence vectors, and second-order co-occurrence vectors (i.e. context vectors that will be discussed in detail below). In most cases co-occurrence represents the similarity between any pair of terms in text corpora. It is assumed that there is an inverse relationship between semantic similarities (e.g. Cosine, Jacard, Dice, etc) and the term-term distance (like the Euclidian distance). This assumption has been applied in most clustering-based text analysis research, such as SenseClusters in [Pur04b].

In the first order representation [Ped97a], [Kul05a], values like unigrams⁶ (BOW), bigrams and multi-grams are considered as the feature space, and the text is normally represented only by the assigned value (binary or frequency based), which is explicitly about the existence of the features. In this case, since most lexical features occur only a few times in each context, if at all, the representation vector tends to be very sparse. It is sometimes worthwhile to apply Singular Value Decomposition (SVD) [Tre97], [Dem90], to reduce the dimensionality of the word-document matrix and the corresponding second

³ By co-occurrence we mean occurrence of a feature in a context explicitly or implicitly (indirectly) via occurrence of another related feature.

⁴ In first order text representation, the text is represented by the set of words that either directly occurred in the text or frequently co-occurred with the earlier group in the corpus; however in second order text representation the text is represented indirectly, which means first, we need to have a vector representation for each of the words in the text, and then the text will be represented as a vector by averaging of the above word representation vectors.

⁵ The word is supposed to be disambiguated over its several meanings or senses.

⁶ Unigrams are single terms which occur more than once, Bigrams are ordered pairs of words, and Co-occurrences are simply unordered bigrams.

order document representation vectors, since SVD can compress a sparse matrix by combining related/dependent columns and eliminating the noisy ones.

Using the first order co-occurrence representation, by simply looking at the vectors we can see which features directly contributed to the contexts. This method has two disadvantages: first, very similar contexts may be represented by different features in the space. Second, in short instances we will have too many zero features for machine learning, such as supervised (classification) or unsupervised (clustering).

In 1998, Schütze [Sch98a] proposed a more integrative context representation method called second order co-occurrence context representation, which is a more integrative method.

In the second order context representation we normally build a word-by-word co-occurrence matrix (over the whole corpus in our case), in which each row represents the first word and the columns represent the second word of the any bigram or co-occurred feature in a corpus [Ped06b]. If the features are bigrams (i.e. we consider their appearance order in the context), the word matrix is asymmetric; for co-occurrences (i.e. we disregard their order in the context) it is symmetric. In either case, the cell values indicate how often the two words occur together, or contain their log-likelihood score of association. This matrix is large, though it could be sparse if it was built over one document or a small number of short texts, since most words do not co-occur with each other. Again, there is an option to apply SVD to this co-occurrence matrix to reduce its dimensionality; not for our representation, however, which tends to reduce the dimensionality in a new way in different levels. Each row of this matrix is a vector that represents the given word in terms of its co-occurrence characteristics. The second order

representation of a context is built up by the average of co-occurrence vectors (over the entire corpus) of its containing words. Instead of averaging, other aggregate functions could be applied (e.g. Maximum and Minimum). In this way, we create a single vector that represents the overall context. For contexts with target words, we usually put a limit on the number of words around the target word that can be averaged for the creation of the target word vector (see details in Chapter 3.).

In second-order co-occurrence representation, two terms that do not co-occur will have similarity if they co-occur with a third term. This is similar to the relation of friend-of-a-friend in social networks [Mil91]. Synonyms are a special example of this kind of relationship. Although they tend not to co-occur simultaneously in a short text, synonyms tend to occur individually in similar contexts, and with the same neighboring words. This method helps reduce the sparseness problem of the data.

Second order co-occurrence has been applied previously for a variety of unsupervised purposes [Ped97a], [Pur04a], [Kul05a], [Ped06b]. We are the first to apply a soft augmented version to a supervised text analysis task, which will be explained in the following chapters. We will specifically describe and implement a contrast parameter, which can be helpful for representations in different tasks with different targeted conceptual levels.

Kulkarni and Pedersen showed that second order context representation is more effective than the BOW on a limited volume of input data or localized scope [Kul05a]. They believed the high level of sparseness over the first-order representation does not provide enough discriminative information for a classification task, due to many zero values for some dimensions in the vector space. The second-order co-occurrence

representation not only contains non-zero values for the occurred features (words/terms) of each context, but also contains many non-zero values for the second order co-occurred features. Therefore, the feature-by-feature co-occurrence matrix, and any corresponding context representation, is less sparse than BOW and the first-order representations.

On the one hand, when data is limited and sparse, exact features (e.g. in the BOW representation) rarely occur in the same role in training and testing data. On the other hand, the second order co-occurrence captures and applies the indirect relations between features as well, and thereby provides more information in order to increase the system's discriminative power.

Previously, this method was applied mainly for unsupervised learning tasks like word sense disambiguation in a given context [Ped98a], [Ped06a], [Pur04a] or short text/context clustering based on specified topics or subjects [Ped05a], [Ped06b], [Ped07].

2.2. Dimensionality Reduction

In most cases, when dealing with text data, dimensionality reduction is considered in order to deal with the computational complexity issue or for smoothing, which is applied to more easily describe the data or fill in missing points [Chu90], [Gre94], [Sch92]. Hence, for any text representation method, the first task is to find a way to reduce the dimensionality⁷.

For a smooth transition to better-known and lower dimensional representative data, two strategies can be used: selecting a subset of the data dimensions (feature selection), or projecting the data into lower dimensional spaces. A rational approach would be to

⁷ Generally speaking, at this level we are dealing with thousands of inherent features which greatly increase the computational complexity of the next steps of text processing.

initially verify if the data is truly high-dimensional data, or if it can be represented in a lower dimensional space without losing the discriminative information. Usually the targeted information can be represented in a low dimensional subspace of the main high dimensional data. This means that in most cases we need less than D (*number of Dimensions*) free variables to uniquely identify the points in our space. Therefore, we should perform feature selection and ignore the extra dimensions, or project our data into the low dimensional space. In order to project the data into a lower dimensional space, we should choose a strategy which minimizes the loss of information and the inevitable distortions [Val08].

In both cases, the first step would be to determine *the effective dimensionality of the data* [Cam03][Lev05][Pet79]. Then we can use the knowledge inside the data to find, and remove, highly correlated dimensions; if this is not achievable, we can attempt to reduce dimensionality using common methods, such as Principal Component Analysis (PCA) or Multi-Dimensional Scaling (MDS). It is essential to monitor closely, since any improper procedure of dimensionality reduction can cause the disappearance of existing patterns hidden in the data [Val08].

Dimensionality reduction, as a pre-processing step to determine a more compact and relevant document representation, can be done either through the various feature selection algorithms, or by extracting some new dimensions and transforming to the new space of these extracted dimensions such as non-negative matrix factorization [Xu03], matrix factorization approaches such as random projections [Bin01], or the well-known Latent Semantic Indexing (LSI) method [Dee88].

Latent Semantic Indexing/Analysis (LSI/LSA) is an indexing method based on the Singular Value Decomposition (SVD) technique, with a solid mathematical background. This method is used to identify some hidden relationships between the terms of a collection of text in a corpus. LSA applies a term-document matrix over a corpus, which shows the occurrences of terms in documents (not word co-occurrence). As the occurrences are considered within documents, the term-document matrix is sparse, with rows showing the terms and columns showing the documents.

After building the occurrence matrix, LSA tries to compute a low-dimensional approximation of the term-document matrix, in order to reduce sparseness and the computational complexity of any further processing on the matrix [Dum05].

Assume that X is a matrix, and the value of element (i,j) describes the occurrence of term i in document j using some well-known method (e.g. tf-idf). Then X is a matrix like:

$$\mathbf{t}_i^T \rightarrow \begin{matrix} & \mathbf{d}_j \\ & \downarrow \\ \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{matrix}$$

Each row of the matrix is a vector representing one of the terms inside the corresponding corpus, which represents its relation to each document:

$$\mathbf{t}_i^T = [x_{i,1} \quad \dots \quad x_{i,n}]$$

Similarly, each column in the matrix is a vector representation of one document in the corpus in which we can see its relation to each term of the corpus⁸:

⁸ Equations and most of the contents of this section have been extracted from the Wikipedia online encyclopedia.

$$\mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

The correlation between the corpus terms can be calculated by the dot product $\mathbf{t}_i^T \mathbf{t}_p$ between the two term vectors (i,p) . In order to have correlation between all pairs of terms in the corpus, we must create the XX^T matrix. With the same logic, since the $X^T X$ matrix contains all dot products between the document vectors, it can give us correlations between pairs of documents in the corpus. It is also a symmetric matrix ($\mathbf{d}_j^T \mathbf{d}_q = \mathbf{d}_q^T \mathbf{d}_j$).

At this point, in order to calculate the singular value decomposition (SVD) of X , we need to find two orthogonal matrices (U and V) and one diagonal matrix (Σ) while:

$$X = U \Sigma V^T$$

Based on the above matrices U , V and Σ , the term and document correlation matrices would be decomposed to:

$$\begin{aligned} XX^T &= (U \Sigma V^T)(U \Sigma V^T)^T = (U \Sigma V^T)(V^T \Sigma^T U^T) = U \Sigma V^T V \Sigma^T U^T = U \Sigma \Sigma^T U^T \\ X^T X &= (U \Sigma V^T)^T (U \Sigma V^T) = (V^T \Sigma^T U^T)(U \Sigma V^T) = V \Sigma^T U^T U \Sigma V^T = V \Sigma^T \Sigma V^T \end{aligned}$$

Since $\Sigma \Sigma^T$ and $\Sigma^T \Sigma$ are diagonal, U contains the eigenvectors of XX^T , and V must be the eigenvectors of $X^T X$.

Assuming the non-zero elements of $\Sigma \Sigma^T$ and $\Sigma^T \Sigma$, the products have the same non-zero Eigen values in the following matrix decomposition:

$$\begin{array}{c} X \\ (\mathbf{d}_j) \\ \downarrow \\ \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \end{array} = (\hat{\mathbf{t}}_i^T) \rightarrow \begin{array}{c} U \\ \left[\begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_l \end{bmatrix} \right] \end{array} \cdot \begin{array}{c} \Sigma \\ \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \end{array} \cdot \begin{array}{c} V^T \\ (\hat{\mathbf{d}}_j) \\ \downarrow \\ \left[\begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_l \end{bmatrix} \right] \end{array}$$

The values $\sigma_1, \dots, \sigma_l$ are the singular values, and $u_1, \dots, u_l$ and $v_1, \dots, v_l$ the left and right singular vectors. Since the only part of U that contributes to $\mathbf{t}_i$ is the i 'th row, we call this row $\hat{\mathbf{t}}_i$. (Likewise $\hat{\mathbf{d}}_j$ for the j 'th column of the V^T). These vectors are *not* the eigenvectors, but have dependency to *all* the eigenvectors. Now by selecting the k largest singular values, and their corresponding singular vectors from U and V , we will have the k approximation to X with the smallest error (Frobenius norm).

We then treat the word and document vectors as a lower-dimensional ‘concept space’. The vector $\hat{\mathbf{t}}_i$ then has k entries, each of which gives us the occurrence of word i in one of the k concepts. In the same way, the vector $\hat{\mathbf{d}}_j$ represents the relation between document j and each concept. This approximation can be written as:

$$X_k = U_k \Sigma_k V_k^T$$

Therefore, we can calculate how documents j and q are related (in the concept space) by comparing the vectors $\Sigma_k \hat{\mathbf{d}}_j$ and $\Sigma_k \hat{\mathbf{d}}_q$ (by a similarity or distance measure). And by the same logic, we can compare words i and p by comparing their representative vectors $\hat{\mathbf{t}}_i^T \Sigma_k$ and $\hat{\mathbf{t}}_p^T \Sigma_k$, in the concept space [Sus05].

The other name for the above method is Latent Semantic Analysis (LSA), which is mainly used for extracting some underlying independent dimensions of texts (see Chapter 8.). However, these dimensions are not interpretable according to human intuition.

LSI has shown good performance in dealing with synonymy (different words with the same meaning), but may fail to address polysemy (same word with different meanings).

Another approach for text dimensionality reduction includes some probabilistic co-clustering (soft-clustering) models, such as Probabilistic Latent Semantic Analysis (PLSA), also known as Probabilistic Latent Semantic Indexing (PLSI) in information retrieval. It was introduced by T. Hofmann in 1999, [Hof99].

PLSA is a statistical technique, used primarily for the analysis of co-occurrence terms in text data in a variety of applications in information retrieval, natural language processing, filtering, machine learning and other related areas.

PLSA evolved from Latent Semantic Analysis by applying a robust probabilistic model. Standard LSA is based on linear algebra and reducing the dimensionality using singular value decomposition, while PLSA is a probabilistic method based on mixture decomposition derived from a latent class model.

Assuming occurrence word w in the document d is denoted as (w,d) , and concept or topic (in text) $z \in Z = \{z_1, z_2, \dots, z_k\}$ as a latent variable for the occurrence word w , which associates an unobserved class variable z_i with each of the observations of occurrence of w . Since the cardinality of $Z = k$ is smaller than the number of words in the corpus, z can be considered for the dimensionality reduction purpose.

The probability of each occurrence has been analyzed by Hoffman [Hof99] as a mixture of conditionally independent multinomial distributions:

$$P(w,d) = \sum_z P(z)P(d | z)P(w | z) = P(d) \sum_z P(z | d)P(w | z)$$

It can be observed that in the left side (symmetric) of the above equation, w and d are both inferred from the latent concept z in similar ways (using the conditional probabilities $P(d | z)$ and $P(w | z)$). However, in the right side (asymmetric) of the equation, for each document d there is a latent class z which is chosen conditionally according to $P(z | d)$,

and there is a word to be taken from concept z according to $P(w | z)$ (Figure 1.) and the conceptual space Z , for all $z \in Z$ has lower dimensionality than the original feature space. [Kim08], [Ami07].

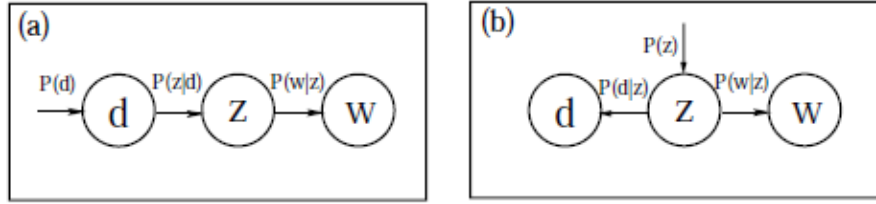


Figure 1. Graphical model representation of the aspect model in asymmetric (a) and symmetric (b) parameterization⁹.

Latent Dirichlet Allocation is another probabilistic technique for dimensionality reduction of text data [Ble03].

PLSA and Latent Dirichlet Allocation could successfully manage the task of topic discovery, as shown by Hofmann in 1999 [Hof99] and Griffiths & Steyvers in 2004 [Gri04].

In 2003, Blei et. al. showed that the above probabilistic model can cause severe over-fitting problems when used for latent semantic analysis [Ble03]. The other problem with PLSA can occur when the number of parameters increases linearly with the number of documents. In addition, though PLSA performs well over the documents in the training set, its model cannot be projected properly over a new set of test documents.

In 2010, Pessiot et. al. introduced an extended version of the (PLSA) model, in which an extra latent variable allows the model to co-cluster documents and terms simultaneously [Pes10].

⁹ Has been extracted from, “PLSI Utilization for Automatic Thesaurus Construction” by M Hagiwara et. Al; Natural language processing, IJCNLP 2005

Recently, co-clustering approaches, which are based on clustering documents and words simultaneously, have been applied to different applications, such as the bipartite graph framework [Dhi01] and the matrix factorization framework [Ban07].

Using the concept based representations for text categorization is another approach to represent text in lower dimensionality. Sahlgren and Coster showed that the performance of the Support Vector Machine can be improved by combining Bag of Concepts (BOC) [Tac05] to conventional BOW representation [Sah04]. They extracted BOC coarse grain features from the original BOW features, either by using some clustering methods, such as distributional clustering [Bak98], or by using factor analytic methods such as singular value decomposition. In 2008 Huang et. al. utilized Wikipedia to create a bag of concept representation, in which each concept is associated to a Wikipedia article in order to exploit the semantic relatedness of the Wikipedia concepts for supervising clustering [Hua08].

Occasionally the data is not truly high dimensional, or we do not need to deal with the full complexity of such spaces. In these cases, looking at the correlation between point distances and some other features of interest may be sufficient.

2.3. Definitions of Ontology in Knowledge Representation

In Artificial Intelligence (AI), Knowledge Representation plays an essential role [Rus03], [Poo98], [Lug04], [Nil98]. In different applications of AI, including semantic web, bioinformatics, enterprise book marking, library science, information architecture and enterprise architecture framework, ontologies are formal means of knowledge representation about the entire domain, or a part of it. An *ontology* defines a formal representation of shared concepts within a domain [Gru93] and their relations. The

ontology normally expresses the implicit properties of that domain or, in a more general way, can model the domain by providing a *new type of shared vocabulary* [Fre08]. We will refer those shared vocabulary in the future as *Conceptual Elements*.

Unlike the defined vocabulary of the domain, which normally appears in the common vector representations, this new vocabulary plays its role in a hierarchal¹⁰ representation of the domain, with different *granular* representativity.

In theory, an ontology is a "...formal, explicit specification of a shared conceptualization." [Gru93]. An ontology provides shared attributes which can be applied to model a domain, and most of the types of objects and/or concepts that exist and govern the domain, in addition to their properties and relations [Arv08]. In other words, an ontology is a perfect representation of "what exists" [Sim87], [Rus03].

There has been much discussion in the literature of information retrieval and knowledge representation regarding what exactly an 'ontology' is, "...however there is a common core that virtually can be applied through the all approaches:

- A vocabulary of terms that refer to the things of interest in a given domain;
- Some specification of meaning for the terms, [ideally] grounded in some form of logic.

" [Usc04]. We translate it to mean a representation of the concepts and inter-relations between those conceptual elements within a domain.

An ontology is a useful way to know the overall definition of the properties and the attributes of the domain.

In information science, the most general ontologies are usually called "upper ontologies" [Nil03], and the main attribute of an upper ontology is to express the extensive semantics

¹⁰ Multi-level representation

(topics) which are applied across a large number of *specific ontologies*. Specific ontologies are recognized under the upper (fundamental) ontology in the ontological hierarchy. This hierarchy could either be designed, based on some regulations, rules and theorems, or extracted (learned) as the *implicit relations among major attributes* of the domain. The upper ontologies usually represent the general entities which cannot be properly represented by any of the lower ontologies, or specific details in the feature set. A fundamental ontology (upper ontology) represents very general concepts of the domain, which are also commonly repeated across the entire domain [Mas02a]. However, in order to model a specific part of a domain, or even an entire domain, we need to explore its innate domain-specific ontologies.

In Natural Language Processing (NLP), an ontology could represent some meanings or senses of terms which apply in a certain domain. For example, the terms *shoe* and *pad* have different meanings; hence, an ontology in the *clothing industry* (as a domain) would represent these words differently than the representation which maps *shoe* and *pad* in the *car industry* (another domain). Some ontologies represent concepts in very specific, and often filtered, ways, which are only fit and valid over the corresponding specific domain, and cannot be projected over any other existing domains inside a corpus (e.g. specific ontologies of irrigation and pesticides under the fundamental ontology of agriculture). Thus, it is plausible to find more than one specific ontology (within the entire corpus) which, although they govern some portions of the corpus, are incompatible over the other portions of the same corpus. In some cases, there are even some ontological relations which cannot be projected over a large number of texts in the same domain, simply because they are too specific. Therefore, in order to represent some more general

concepts over a wide range of documents which are only different in specific details, there is a need for an automatic functionality, to merge specific ontologies into some upper ontology. Following the above example, although it is important to know what our texts are about (e.g. harvesting, irrigation, pesticides, siloing, pedology, etc.), at the same time it could be essential to recognize that all of them are about 'agriculture'.

According to the above, a fundamental ontology (Upper Ontology in Figure 2.) helps us to model the common objects which can generally represent a wide range of specific domains, and their specific ontologies, at a more general level. In other words, upper ontologies can be considered as representations of intended meanings; this is an empirical approach to extract the topics in a given domain.

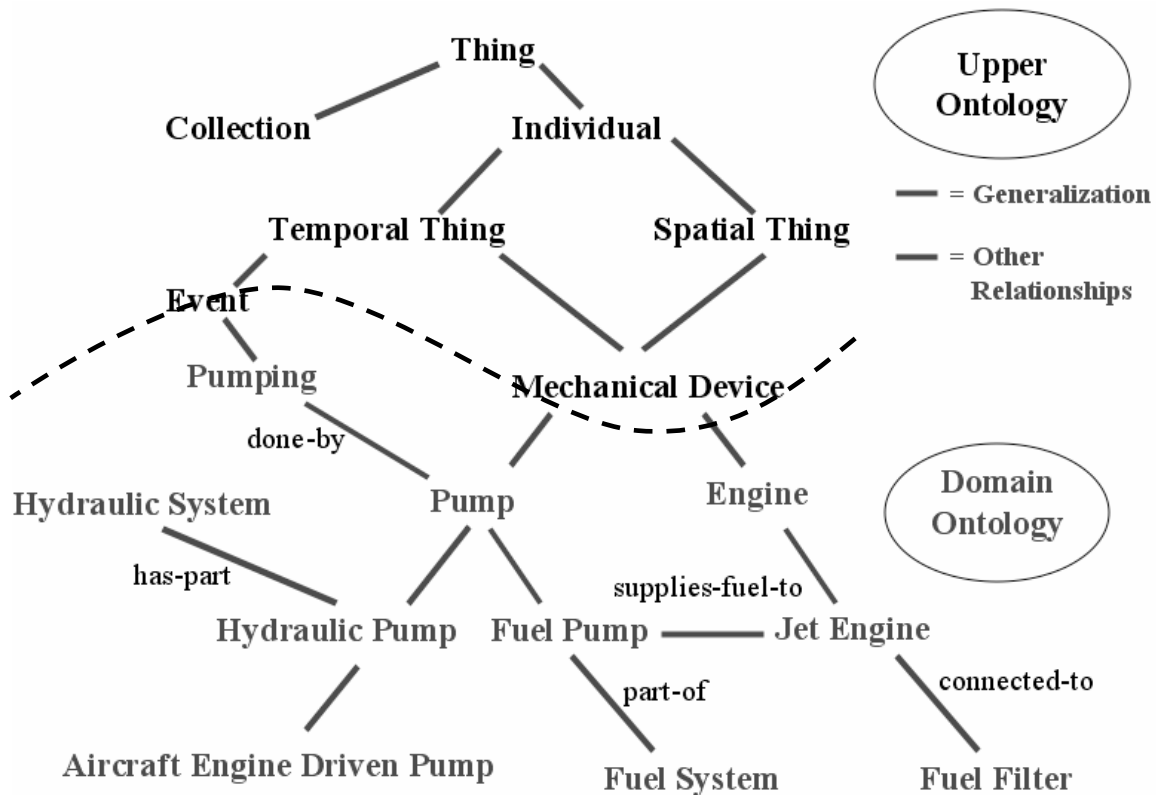


Figure 2. An example of ontology [Usc04]. The dashed line indicates the approximate possible border between the Upper Ontology and Domain Ontology.

Ontologies represent a variety of attributes over the entire corpus. However, when the goal of the learning task is considered, there could be some attributes—and, consequently, some specific part of the ontology—which do not carry any useful information for the targeted goal of learning, which is imposed by the class attribute values. For instance, in the above example of “pad” and “shoe”, if the class attribute values are expressing some entities in the car industry as the targeted goal of learning, then even if there is/are some scattered attribute(s) of the clothing industry in the corpus, these attributes would not carry useful information in our representation, and would only increase the perplexity of the learning algorithm. Thus, the ontology parts which do not represent any helpful aspects of the domain for the learning goal should be removed from the representation. Moreover, since there could be more than one issue to be addressed in some contexts, it is possible to have more than one ontological attribute rooted in different aspects of the domain which is represented.

2.4. Generality and Granularity of Knowledge

The concept of information granulation was first introduced by Zadeh in 1979 [Zad79] through fuzzy sets. The main ideas of information granulation can be found in machine learning, databases, quantization, the theory of belief functions, interval analysis, divide and conquer, rough set theory, cluster analysis, and others [Zad97]. The evolution of ideas in Granular Computing (GrC) can be tracked in different applications [Paw98], [Pol98], [Sko98], [Yag98], [Zho99], [Yao00], [Hir01], [Ped01], [Yao02], and include clusters of a universe, subsets, and classes as major ingredients [Yao00], [Zad97]. Granulation of the universe, relationships between granules, description of granules, and

computing with granules are some of the essential characteristics of granular computing, which can be studied from the aspects of either the *construction of granules*, or *computing with granules*.

Granular computing was applied in the representation, formation and interpretation of granules, and continued in the utilization of granules in problem solving. It can also be considered as a matter of algorithmic and semantic point of views [Yao01]. Semantic studies are mostly concentrated on questions of “Why?” such as, “Why are some objects considered in one granule?” and “Why are some granules related to each other?” Algorithmic works are focused on questions of “How?” (i.e. detailed aspects of the creation of information granulation, and computing with them).

Granules are usually defined and constructed based on the generalized constraints concepts. Zadeh writes: "There are three basic concepts that underline human cognition: granulation, organization and causation. Informally, granulation involves decomposition of the whole into parts; organization involves integration of parts into the whole; and causation involves association of causes with effects. Granulation of an object A , leads to a collection of granules of A , with a granule being a clump (*cluster*) of points drawn together by indistinguishability, similarity, proximity or functionality." Furthermore, he states: "Modes of information granulation (IG) in which the granules are crisp (c-granular) play important roles in a wide variety of methods, approaches and techniques. Crisp IG, however, does not reflect the fact that in almost all of human reasoning and concept formation the granules are *fuzzy* (f-granular)." This is explained in more detail in [Zad97]. Granules are normally labeled with natural language words or fuzzy sets, and the relations between different granules are usually represented by fuzzy graphs or fuzzy

logic rules. Many more granular computing models have been introduced, by authors such as [Lin98], [Paw98], [Ped01], [Sko98] and [Yag98].

In order to clearly express the role of different granularity levels in a text, and its relation to knowledge extraction from a text, we will now introduce the specific meaning of Ontology in Knowledge Representation as a base of our algorithm to extract the knowledge and map the learning model, through the fundamental to specific stacks of lightweight ontological levels of a contextual domain.

2.5. Definition of the Light-weight Ontology

In many practical applications, ontologies are introduced as taxonomic formal structures of simple or compound terms, which are assigned to their definitions. However, in some semantic web ontologies there are so-called *lightweight ontologies*, which are used to automatically represent semantic relationships among vocabularies, to assist with content-based access to the Web data in certain communities [Mas02b].

Since web directories like Google or Yahoo are not represented in a formal language, which can be used for *automating reasoning*, they are considered *lightweight ontologies* (see Figure 3.) [Usc04]¹¹. Ulrich Reimer defines the lightweight ontology as “a collection of concepts which are related with each other via associations that are un-typed and do not specify of what kind the relationship is”.

In these cases, the intended classification hierarchies are latent in natural language contexts (in contrast to fully-defined ontologies), and each new entry can be predicted

¹¹ For a better understanding of the meaning of ontology that was used for this dissertation, we recommend you review [Usc04].

based on the members of the community, or the community type (or other types of background knowledge). The applied ontological approach in this dissertation is a *lightweight ontology* that, for simplicity, we call “*Ontology*”.

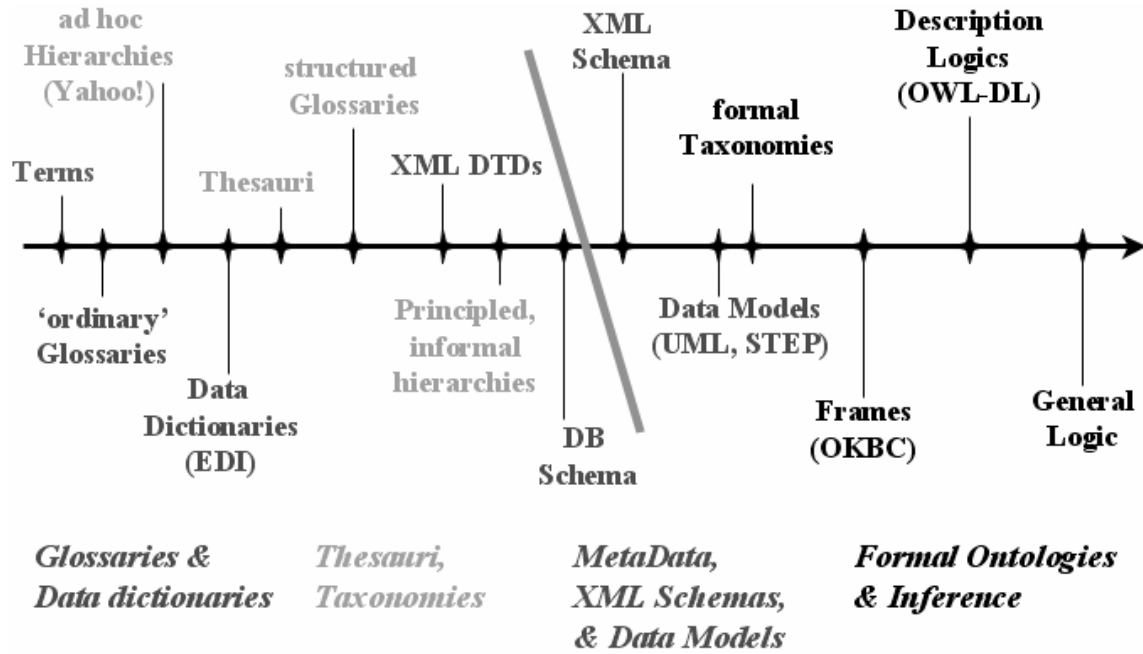


Figure 3. Types of Ontology [Usc04]. The diagonal line indicates the approximate border between formal ontology and lightweight ontology.

Different approaches to ontologies are essentially based on different definitions of terms, and this can provide a kind of continuum illustration for a variety of ontologies. At one extreme, we could find very lightweight ontologies that consist of terms only, with little or no specification of the meaning of the terms. At the other end of the continuum, however, we might observe rigidly formalized logical theories which comprise the ontologies (Figure 3). As we move from left to right along the continuum, the amount of meaning specified and the degree of formality increase, and, consequently, the degree of ambiguity decreases[Usc04].

2.6. Clustering

In this dissertation, to overcome the curse of dimensionality that potentially comes with the SOSCO representation, instead of including the entire feature space in our representation, we can introduce some expressive representatives for each group of close enough¹² features (conceptual elements) as a cluster, and then go through the dimensionality/ambiguity reduction (See section 6.2. for more details). For this purpose we need to choose a clustering method that can perform effectively on data with high dimensionality. Text clustering has been used for a variety of concept analysis purposes, from topic/subject detection to term sense disambiguation [Ped05a], [Ped06b], [Ped07] [Ped98a], [Ped06a], [Pur04a].

Figure 4 illustrates a two level hierarchal clustering over a small vocabulary about Victoria province¹³. The main topic is in the center, with a circle around it. Related subjects and sub-topics are in the smaller circles around the main circle, and some related words from the domain vocabulary are connected to these circles.

¹² Identifying enough closeness is related to the target concept to be addressed by the representation (i.e. enough closeness rates would be different when the target concept is agriculture, rather than if the target concept is something more specific, such as irrigation). However, identifying the proper size of an expressive cluster is a major task of the TOR-FUSE algorithm, and will be discussed later.

¹³ Excerpted from: "http://web2.uvcs.uvic.ca/elc/sample/intermediate/wt/wt_2b.htm"



Figure 4. An Example of Subject/Topic clustering.

It now seems necessary to review the well-known clustering algorithms, in order to find some specific methods tailored for high dimensional data. Using any clustering method implies having distinct knowledge about the variety of distance measures that can potentially be applied through the clustering algorithms. Hence, we briefly discuss these metrics first, and then examine the clustering approaches.

2.6.1. DISTANCES IN HIGH DIMENSIONAL DATA

The high dimensionality of the space hinders efforts to define a proper measure for choosing or searching for the nearest neighbor points. One problem with high-dimensional data is that we cannot visualize data properly in high dimensionality ($>3D$).

The high dimensional spaces have some attributes which contradict our intuition, and that makes it difficult to envision a clear and rational image of the data. Hence, it would be

problematic to apply our intuition directly in order to understand and tackle problems in high dimensional data. These particular properties are collectively known as 'the curse of dimensionality' [Val08]. The instability of the determination of these points becomes more difficult as the number of dimensions increases. Consequently, this affects the high dimensional modeling in any type of prediction task.

To illustrate the "vastness" of a high-dimensional Euclidean space, we can compare the proportion of the volume of a hyper-sphere with d dimensions and radius r , to the volume of a hypercube with sides of length $2r$, and equivalent dimensions. In his paper, Koppen shows that the ratio of the volume of a hyper-sphere to the volume of its embedding hypercube goes to zero by the order of $n!$, in which n is the number of dimensions. [Kop00]. In other words, nearly all points of a high-dimensional space are "far away" from the center; or, put another way, the high-dimensional space unit consists entirely of the "corners" of the hypercube, with almost no "middle" [Bel57] (Figure 5.).

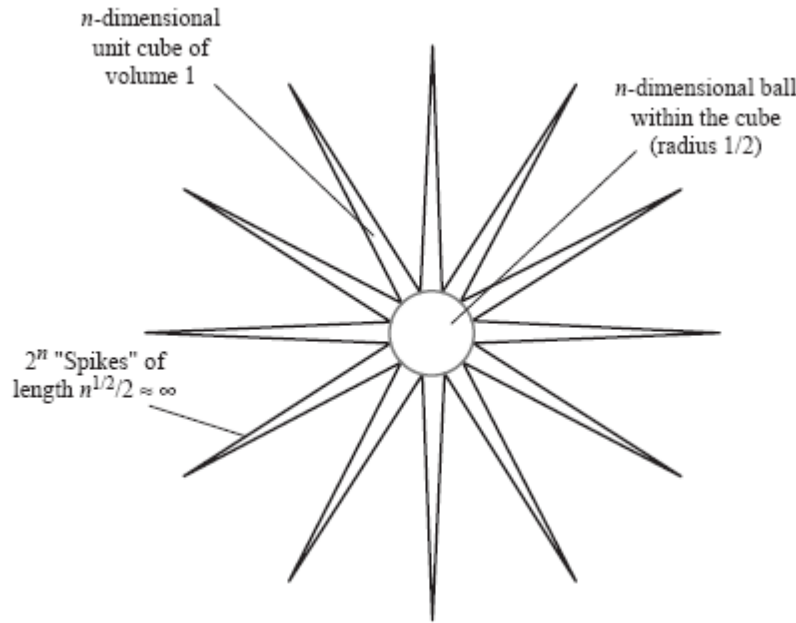


Figure 5. Spiking Hyper-cube [Hec91]

Since n goes to infinity, the curse of dimensionality is considered a significant obstacle to defining a meaningful distance in high-dimensional space, specifically when we are dealing with a limited (low) number of points/samples (e.g., in the nearest neighbourhood algorithm). For example, for any two randomly selected points in a hypercube, their Euclidian distance will be nearly identical when n increases [Kop00]. Hence, the discrimination between the nearest and the farthest pair of points becomes meaningless [Kri09]:

$$\lim_{d \rightarrow \infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \rightarrow 0$$

Since the Euclidian distance between any two points in a given dataset converges, for such data the concept of distance gradually becomes imprecise, which would be a considerable problem in most clustering algorithms.

Although, in general, we cannot find a perfect distance definition, depending on the problem to be modeled we can select a better distance function, and try to overcome or minimize the distance concentration phenomenon.

There are various distance functions that can replace the **Euclidean distance** measure.

1- The **Minkowski distance** is the generic form of most of the other well-known distances, and it is defined as follows:

If A and B are two points, where $A, B \in \mathbb{R}^n$; then we have:

$A=(x_1, x_2, \dots, x_n)$ and $B=(y_1, y_2, \dots, y_n)$ and the generic Minkowski distance D_m is calculated as:

$$D_m(A,B) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}.$$

However, the Minkowski distance is equal to the Euclidean distance when $p=2$, equal to the Manhattan or City-block distance when $p=1$, and is even equal to Chebyshev distance, but only when $p \rightarrow \infty$ [Dez06] as in this case:

$$D_m(A,B) = D_c(A,B) = \max_{i=1..n} |x_i - y_i|$$

Essentially, the generic Minkowski distance is computed by moving along the coordinate axis [Fra07].

2- Another interesting distance function is the well-known **cosine distance**, which is a popular norm in the text mining community [Sal83], [Sal88]. Originally, in this norm every text has an associated vector of word frequencies, and the similarity between texts is calculated based on the dot product between these vectors. In addition to the Minkowski distance, we can use a slightly modified definition of the cosine similarity [Val08] that produces a distance in the $[0 \dots 1]$ interval over the Second Order Soft Co-Occurrence (SOSCO) representation¹⁴ vectors as follow:

$$dist(\vec{x}, \vec{y}) = \frac{1}{2} \left(1 - \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|} \right)$$

3- The **Mahalanobis distance** measure, introduced by P. C. Mahalanobis [Mah36], is another well-known distance function. It is calculated by considering the correlations between features in the feature space. In this measure, the correlation between any pair of features contributes to the calculated distance, in which each feature participates according to its relationship with the other features.

¹⁴ Will be explained in the next chapter.

If $\vec{x}=(x_1, x_2, \dots, x_n)^T$ and $\vec{y}=(y_1, y_2, \dots, y_n)^T$ are two points in n dimensional feature space, and S is the correlation/covariance matrix of the features, the Mahalanobis distance is calculated as:

$$D_m(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}$$

When the covariance/correlation matrix S is diagonal, it becomes the *normalized Euclidean distance*¹⁵:

$$D(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^n \frac{(\vec{x}_i - \vec{y}_i)^2}{\sigma_i^2}}$$

where D is the standard deviation of x_i over the given data points belonging to R^n .

Similarly, if $\vec{x}=(x_1, x_2, \dots, x_n)^T$ in a distribution D with a range of values with mean $\vec{\mu}=(\mu_1, \mu_2, \dots, \mu_n)^T$, and S is the correlation/covariance matrix of the feature space, the Mahalanobis distance of the vector $\vec{x}$ from the mean (centroid) of D is calculated as:

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}$$

However, the regular Euclidian distance $D_E(\vec{x})$ is computed as:

$$D_E(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T \cdot (\vec{x} - \vec{\mu})}$$

and does not take correlations into account.

We recalled the problem of estimating the distance of a given point in n -dimensional space to the center of a cluster. Mahalanobis' approach gives us an applicable average (cluster center) of the cluster points, in order to determine a quantitative sense of the

¹⁵ The proof can be found at: De Maesschalck, R.; D. Jouan-Rimbaud, D.L. Massart; The Mahalanobis distance. Chemometrics and Intelligent Laboratory Systems 50 (2000) :1–18

center point (centroid). However, this point might or might not be a real point in the cluster.

Since we need to choose a proper clustering method to extract the conceptual elements in the next step, we decided (inspired by the Mahalanobis method) to take into account the inter-relationship of the words as dimensions (feature space) of the textual space for better representation of a text in our SOSCO representation, in order to obtain a meaningful distance measure.

2.6.2. HIERARCHICAL CLUSTERING¹⁶

Hierarchical algorithms create a hierarchical structure of clusters, in which each cluster can either be divided into two successive clusters or merged with another cluster to create a new, more general cluster.

These algorithms construct either an *agglomerative (bottom-up)* hierarchy, in which clusters at the leaves include only one instance and are consecutively merged into larger/general clusters, or a *divisive (top-down)* hierarchy, with one general cluster that recursively splits into smaller/specific clusters.

The hierarchical structure of these sorts of clustering methods can be represented as a tree, with the root as a single cluster including all instances, and the leaves as groups of small numbers of instances that have high similarity in each group.

¹⁶ For more details about clustering methods refer to: "Review of Clustering Algorithms" by Wesam Ashour Barbakh, Ying Wu and Colin Fyfe; *Studies in Computational Intelligence*, 2009, Volume 249/2009, 7-28, DOI: 10.1007/978-3-642-04005-4_2

'Pruning' the tree to a certain level will result in clusters with a relative level of precision.

As a measure of similarity between pairs of instances, we can apply a non-negative-valued similarity function, or use the well-known distance functions. However, there are different linkage criteria between pair of instances that determine if a cluster should be split in two or be merged with the other one.

The linkage criteria between pair of elements normally include:

- Single-linkage clustering: according to the minimum distance between instances of each cluster;
- Complete-linkage clustering: according to the maximum distance between instances of each cluster; and,
- Average linkage clustering: according to the average/mean distance between instances of each cluster.

The ending condition criteria of a hierarchical clustering algorithm could consist of one or more of the following:

- Minimum intra-cluster variance: according to the minimum summation of intra-variance values of the entire cluster (an optimization problem);
- Certain intra-cluster variance: where the intra-cluster variance cannot be more than a given value;
- a certain number of cluster criterion; and/or,
- Certain inter-cluster distance criterion.

2.6.3. PARTITIONAL CLUSTERING

In partitional methods, all clusters are normally created randomly at the beginning, and then modified iteratively. However, the methodology can also be applied to the divisive algorithms in the hierarchical clustering. For example, the k -means algorithm starts with each instance assigned to a cluster (for the first number of instances equal to the given number of clusters), then continues to assign the remaining instances to the cluster whose mean (centroid) is closest; 'closest' is defined according to an already defined distance function. The algorithm will modify the clusters by re-calculating the cluster means and reassigning the instances iteratively, and ends when some convergence criterion is met (typically when the assignment hasn't changed).

Partitional algorithms are normally faster than hierarchical methods, allowing them to run on large data. However, since the first clusters are assigned randomly, the final results could change from run to run. For cases in which the mean cannot be defined, the k -medoids method could be applied, where the medoid is a real instance in the cluster that is closest to the average of the cluster members.

2.6.4. FUZZY C-MEANS CLUSTERING

Fuzzy clustering methods assign each instance of data a degree of belonging to each cluster (from 0 to 1), instead of assigning it solely to one cluster. For example, in the fuzzy k -means algorithm the mean of each cluster is the mean of all the data points, weighted by their degree of belonging to each cluster. The degree of belonging is calculated according to the inverse of the distance to the cluster centroid. Then the

coefficients (degrees of belonging) are normalized and 'fuzzyfied', so their sum is equal to 1. The results depend on the initial setting of the coefficients.

There are also algorithms that provide the option of choosing one of the above simple methods as core, and boosting their performance by imposing restrictions or extra conditions. These algorithms are mainly applied to high-dimensional and/or very large datasets (e.g. text and bio-information).

According to Kriegel, Kröger & Zimek [Kri09], for proper clustering of the feature space there is an inherent problem, which they call High-dimensional Clustering problem. Any clustering method tends to cluster the related points, according to their attributes' values; thus, when the number of attributes grows significantly, there could be attributes whose values are not meaningful for a given cluster. Hence, we will briefly review them, in order to choose an appropriate algorithm for our text clustering purposes.

2.6.5. SPECTRAL CLUSTERING (KERNEL PRINCIPAL COMPONENT ANALYSIS)

Spectral clustering methods are typically based on a similarity matrix over the entire data. In these methods, dimensions of the similarity matrix are reduced through a decomposition process (e.g. Eigen value decomposition), or by component analysis. Then a simple clustering method (e.g. *k*-means) would be applied over the data in a lower dimensionality. Examples of this approach include *Normalized spectral clustering* methods, which have been introduced in [Shi00] and [Ng02].

2.6.6. DENSITY-BASED CLUSTERING

In these methods, a region is considered a cluster only if the density of data points is not less than a certain threshold (figure 6.) [Est96]. Density-based clustering algorithms are usually used to explore clusters with an uneven shape. DBSCAN and OPTICS are two typical example algorithms of this approach.

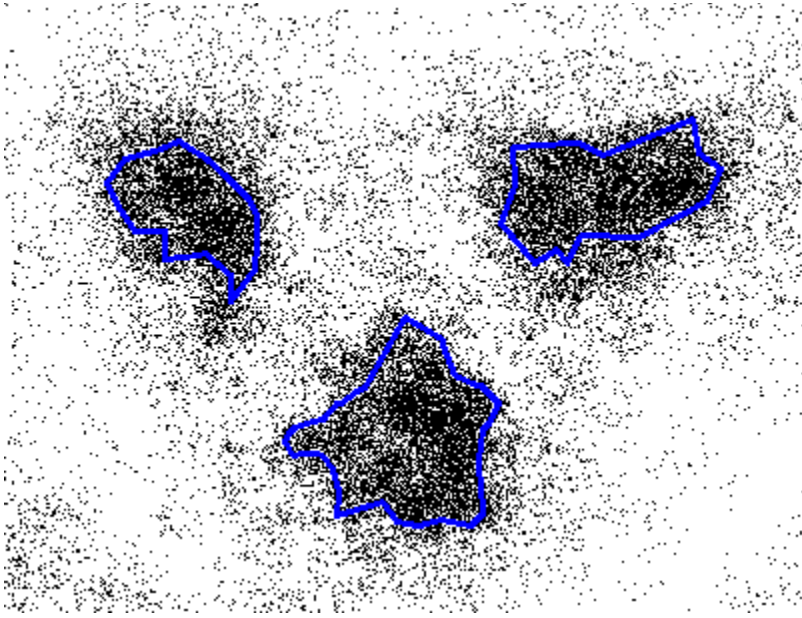


Figure 6. Density-based clustering, encloses and separates clusters using a density parameter.

2.6.7. SUBSPACE CLUSTERING

Subspace clustering methods search for clusters that can be recognized in a subset of the feature space of our data. The greatest advantage of these methods is to disregard (or suppress) the effect of irrelevant attributes, in order to explore a meaningful cluster that emerged from the subset of the features in high-dimensional data. For example, in word clustering¹⁷ there are often some words close enough to potentially create a cluster.

¹⁷ Based on a similarity matrix which is usually symmetric, and where each row/column represents a word (feature) in the high dimensional feature space.

However, there are large number of features (words) in their vector representations with non-considerable degrees of relevancy (high variance according to a certain threshold), which contribute to the similarity/distance measure. These features degrade the level of their similarity and, consequently, prevent emergence of their meaningful cluster. Kriegel called this the local feature relevance problem, which means that different clusters might be found in different subspaces [Kri09]. In a given large number of attributes, there are usually many which are correlated. Therefore, there could be clusters which belong to more than one arbitrarily assigned subspace. Subspace clustering tends to detect all the clusters in the entire combinatory subspaces. In other words, a point can be a member of more than one cluster and belong to a different subspace (Figure 7.). Subspaces can either be axis-parallel or affine. Due to the serious effects of the curse of dimensionality, subspace clustering is normally used synonymous with general clustering in high-dimensional data.

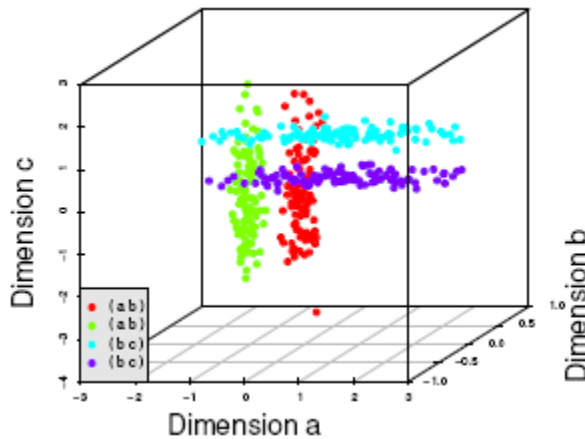


Figure 7. Sample dataset with four clusters, each in two dimensions in which the third dimension is considered as noise. Points from two clusters can be very close together, but this may confuse many traditional clustering algorithms. [Par04]

The image above illustrates a special three-dimensional space, where some clusters can only be identified in one of two possible combinations of three axes.

In this approach to the problem of subspace clustering, the potential number of different $2d$ and $3d$ subspaces of a space with n dimensions can be seriously problematic. Cases where the subspaces must not be only axis-parallel, can create an infinite number of possible subspaces. Therefore, subspace clustering algorithms should apply some kind of heuristics in order to cope with the computational complexity. However, heuristics can, to some degree, compromise the accuracy of their results. For example, creating higher-dimensional subspaces by only combining lower-dimensional ones (downward-closure) means that any subspace of a space that contains a cluster will also contain a cluster. There are some approaches to applying the above heuristic in most of the traditional algorithms, such as CLIQUE [Agr05] and SUBCLU [Kai04].

2.6.8. PROJECTED CLUSTERING

In these algorithms, the general approach is based on applying a special distance function (already listed) into a simple clustering algorithm. DBSCAN was the first density-based clustering method that applied a distance function that assigns more weight to some dimensions, and thereby boosts some sufficiently low difference dimensions in order to efficiently cluster the points [Est96].

As an augmented example of projected density-based algorithms, we recall the PreDeCon algorithm that checks for the attributes which seem to support a clustering for each point, and then modify the distance function so that the roles of dimensions with low variance are amplified in the distance function [Boh04]. The PROCLUS algorithm [Agg99] is another example which uses a similar approach to k-means clustering. This algorithm initially assigns some instances as the means, then for each mean the subspaces are determined by attributes with low variance. Hence, points which are closest to the mean

(considering only the subspace of each mean for calculating the distance) are assigned to the cluster. The algorithm then continues like a regular Partitioning Around Medoids (PAM) algorithm. Although, the distance function assigns different weights to different attributes, it never assigns a zero weight to an attribute (i.e. it never eliminates irrelevant attributes). The algorithm is called a *soft-projected clustering algorithm* [Agg00], [Agr05].

2.6.9. HYBRID APPROACHES

There are some algorithms that although do not assign each point to a unique cluster, no points belong to all the clusters in all the subspaces. In these methods, there are many instances which are assigned to one or more than one cluster, but not necessarily to an exhaustive set of clusters. For example, FIRES is basically a subspace clustering algorithm, however it applies an aggressive heuristic to produce all its subspace clusters [Kri05].

2.6.10. TEXT CLUSTERING

Slonim and Tishby [Slo01] apply the *information bottleneck method* [Tish99] to extract word-clusters that preserve the information about document categories and apply these clusters as a new feature space for text classification task.

In 1998 Baker et. al. [Bak98] uses Distributional Clustering [Per93b] for document classification task. Their technique clusters words based on their class labels distribution associated with each word in a document. They shows that their method can reduce the dimensionality and compress the feature space much more aggressively with better classification accuracy compare to other unsupervised methods such as Latent Semantic

Indexing. In 2001 Bekkerman et. al. [Bek01], [Bek02] applied the distributional clustering technique for feature selection combined with a support vector machine classifier to improve previous text categorization methods. In 2008 and 2009 Huang et. al. presented a concept-based document representation in which they map the terms and phrases within documents to their corresponding articles (or concepts) in Wikipedia. They defined a similarity measure to evaluate the semantic relatedness between concept sets for any two documents and utilized it to cluster the documents in the concept base feature space [Hua08], [Hua09]. Ahmed et. al. introduced an algorithm (Hierarchical SISC) that captures the underlying correlation between each pair of class labels in a multi-label environment where there are more than one single class label can be assigned to each document in a corpus (e.g., Reuters-21578 dataset). Then they define their subspace clustering algorithm based the extracted correlations to be utilize for a fuzzy clustering method [Ahm10].

In 2011 Yan Li et. al. introduced a subspace decision cluster classification (SDCC) model consists of a set of disjoint subspace decision clusters, in which each of the clusters are labeled with a dominant class to determine the class label of a new given instance falling in the cluster [Li 11].

Since a combination of ensemble learning methods will be applied for the final part of our TOR-FUSE algorithm, the rest of this chapter will review ensemble learning.

2.7. Ensemble learning

The idea of ensemble learning is to apply multiple learning algorithms, and then combine their predictions in different ways in order to achieve higher discriminative power. The different ways include bagging, boosting, voting, ranking, stacked generalization, error-

correcting output codes, cascading and more. These are discussed briefly in the following sections.

In 1993, Perrone and Cooper [Per93a] introduced a theoretical solution for ensemble methods, and claimed that they can significantly improve the performance of regression estimating.

In 1995, Cho and Kim [CHO95] applied fuzzy logic in order to combine multiple neural networks, and induce higher levels of accuracy in their results. In the same year, Krogh and Vedelsby [Kro95] showed that the performance of ensembles can be significantly increased when unlabeled data is used during the training process. In 1996, Sollich and Krogh [Sol96] showed the potential of large ensembles, when they are applied over a group of classifiers which normally over-fit the training data. They proved that this combination potentially provides the maximum benefits of the variance-reducing attributes of ensemble learning.

In 1997, Larkey and Croft [Lar97] showed that combining classifiers in text categorization can improve the performance. In 1998, Kittler [Kit98a] introduced a theoretical framework for combining classifiers in a two-way combination method: combining the different models on the same learning task (distinct representations) and combinations of different classifiers on a shared representation. They showed that, for the shared representation, the combination was effective and obtained a better estimation of the class probabilities. In 1999, Opitz [Opi99a] conducted studies on feature selection specifically for ensembles. In the same year, Miller and Yan [Mil99] introduced a critic-driven ensemble of classifications. And in 2000, Jain, Duin and Mao [Jai00] introduced a list for combinations of classifiers with many reasons for combining multiple classifiers,

including different feature sets, different training sets, different classification methods and different training sessions. They also reported that combining any of above may improve the overall classification accuracy.

In 2000, Kuncheva, et al. [Kun00] focused on dependency and independency in the combination of classifiers. Their results confirmed the intuition that a combination of classifiers when the output of one (as added features) can improve the accuracy of the other (as dependent classifier), are better than a combination of independent classifiers (e.g. voting). They also showed that this relationship is reciprocal.

In 2002, Shipp and Kuncheva [Shi02] continued their studies of the relationships between different methods of classifier combinations and measures of diversity in combining classifiers. They showed that the unknown relationship between diversity and accuracy discourages calibration of the diversity.

In 2003, Kuncheva and Whitaker's [Kun03a] results raised doubts about the usefulness of diversity measures for building classifier ensembles in real-life pattern recognition problems.

In the following year, Chawla, et al. [Cha04] constructed a framework for building hundreds or thousands of classifiers on small subsets of data in a distributed environment. Their experiments showed that the approach is fast, accurate and scalable. In 2005, Melville and Mooney [Mel05] proposed 'Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples' (DECORATE), a new framework for generating ensemble learners that directly creates diverse hypotheses using additional artificially-constructed training data. Their results consistently outperformed the bagging and random-forest-based classifiers on relatively large datasets, and outperformed

AdaBoost (see Section 2.7.2) on small training sets, with comparable performance on larger training sets. In the same year, Garcia-Pedrajas, Hervas-Martinez and Ortiz-Boyer [Gar05] introduced a cooperative co-evaluative framework for designing neural network ensembles. And the next year, Reyzin and Schapire [Rey06] reported that maximizing the margin can also increase classifier complexity. They showed that maximizing the margins at the expense of other factors, such as base-classifier complexity, is not useful. In 2007, Canuto et al. [Can07] investigated how the choice of component classifiers can affect the performance of selection-based and fusion-based combination methods. It is notable that the highest accuracies were almost always achieved using hybrid methods. In the same year, Kuncheva and Rodriguez [Kun07] presented a combined fusion-selection method to classifier ensemble design, which they called the 'random linear oracle'. Each classifier in the ensemble is replaced by a mini-ensemble of a pair of sub-classifiers, with a random linear oracle choosing between the two. Their experimental results showed that all the ensemble methods benefited from their approach. In 2008, Leap, et al. [Lea08] conducted a study on the roles of correlation and autocorrelation in classifier fusion and optimal classifier ensembles. Their results showed that fusion methods applying neural networks outperform fusion methods based on Boolean rules. In the same year, Martin Sewell presented an extensive and useful review on Ensemble Learning that has been used for the following sections [Sew08].

2.7.1. BAGGING

In 1996, Breiman [Bre96] introduced Bagging ensemble learners for the first time. In 1998, Ho [HO98] applied a random subspace method, and defined decision random

forests. The method performed well in practice, and worked best when the dataset has both a large number of features and a large number of instances.

In the following year, Opitz and Maclin [Opi99b] compared bagging with two boosting methods, AdaBoost and arching. They showed that, for data with few noise samples, boosting outperforms bagging, but overall bagging is the most appropriate.

In 2000, Dietterich [Die00] conducted a comparison study on the effectiveness of randomization, bagging and boosting for improving the performance of the C4.5 decision-tree algorithm. Their experiments showed that, in situations with little or no classification noise, randomization is competitive with, and slightly superior to, bagging, and that boosting outperforms both. However, in situations with considerable classification noise, bagging is much better than boosting, and sometimes better than randomization.

In 2002, Skurichina and Duin [Sku02] applied and compared bagging, boosting and the random subspace method to linear classification algorithms. They showed that boosting is useful for large training sample sizes, while bagging and the random subspace method are useful for critical training sample sizes; this was very similar to the results of Skurichina's PhD thesis¹⁸ of the same year.

In 2004, Valentini and Dietterich [Val04] focused on bias-variance in SVMs for the development of SVM-based ensemble methods. They proposed two promising approaches for designing ensembles of SVMs. The first is to employ low-bias SVMs as base learners in a bagged ensemble, and the second is to apply bias-variance analysis, to build a heterogeneous set of accurate and low-bias classifiers.

¹⁸ Can be accessed at: <http://ict.ewi.tudelft.nl/~marina/thesis.html>

2.7.2. BOOSTING

In 1990, a theoretical paper by Kleinberg [Kle90] introduced a new general discriminatory method for separating points in multidimensional spaces, by applying a stochastic method called stochastic discrimination (SD). Basically, the method uses low performance algorithms as input, and creates boosted solutions. In 1996, Freund and Schapire [Fre96] introduced the boosting algorithm AdaBoost, and in 1999 Schapire [Sch99] introduced extensions to the AdaBoost algorithm, and explained the underlying theory of boosting. In 2000, Kleinberg [Kle00] provided algorithmic implementation in an attempt to bridge the gap between the theoretical promise shown by stochastic discrimination, and a more practical solution. He also showed that Kleinberg's theory of “stochastic discrimination” outperformed both boosting and bagging in the majority of benchmark problems it was tested on.

In her 2001 PhD thesis, Skurichina [Sku01] dealt with the problem of stabilizing weak classifiers, and compared bagging, boosting and the random subspace methods. She concluded that bagging is useful for weak and unstable classifiers with a non-decreasing learning curve and critical training sample sizes, while boosting is efficient only for weak, simple classifiers with a non-decreasing learning curve created on large-size training data. The random subspace method is efficient for weak and unstable classifiers that have a decreasing learning curve, and are constructed on small and critical training data. In 2008, Zhang and Zhang [Zha08] presented a local boosting algorithm for dealing with classification, based on the boosting-by-resampling version of AdaBoost. Their experimental results showed the efficiency of their algorithm, which was more accurate and robust than AdaBoost.

2.7.3. VOTING

In 1992, Xu, Krzyzak and Suen [XU92] developed some methods for combining multiple classifiers, and applied them to handwriting recognition. They divided the problems of combining multi-classifiers into three types, according to the levels of output information of various classifiers. They continued by comparing three different approaches of the first type: voting, the Bayesian formalism, and the Dempster-Shafer formalism that combines classification predictions according to their degree of belief (probabilities). They concluded that the Dempster-Shafer formalism outperforms the other methods, since it can obtain robust high recognition and reliability rates simultaneously.

In 1995, Lam and Suen [LAM95] conducted a survey on the performance of four combination methods: the majority vote, two Bayesian formulations, and a weighted majority vote (assigned by a genetic algorithm). They concluded "...in the absence of a truly representative training set, simple majority vote remains the easiest and most reliable solution among the ones studied here."

In 1997, Raftery, Madigan and Hoeting [Raf97] offered two augmentations of the Bayesian model averaging: Markov chain Monte Carlo and Occam's window. These two extensions addressed the uncertainty problem of the model in linear regression models.

Then Lam and Suen [LAM97] conducted a survey on majority voting in pattern recognition applications.

The next year, Kittler et al. [Kit98b] developed a combination method which empirically compared the product, sum, min, max and median rules, and majority voting. They showed that the sum rule outperformed other classifier combination schemes and, with

regard to sensitivity analysis, showed that the sum rule is also more effective in error estimation.

In the same year, Schapire et al. [Sch98b] introduced a definition for the effectiveness of voting methods. They assigned this to the distribution of *margins* of the training examples with respect to the generated voting classification rule, where the *margin* of an example is simply *the difference between the number of correct votes and the maximum number of votes received by any incorrect label*.

In 2000, Tax et al. [TAX00] questioned whether to combine multiple classifiers by averaging or multiplying. They showed that averaging works better for estimating the posterior probabilities, when they are not already effectively estimated by the individual classifiers. However, for problems involving multiple classes—even with good estimates of posterior class probabilities—the product combination rule still outperformed the average combination. In 2002, Kuncheva [Kun02] defined formulae for the classification error for the following fusion methods: average, minimum, maximum, median, majority vote, and oracle. This study showed that, for a uniformly distributed posterior probability the minimum/maximum method performed best, while for normally distributed errors the fusion methods all had very similar performance. In 2003, Kittler and Alkoot [Kit03] conducted an extensive study of the ‘sum’ versus ‘majority vote’ in multiple classifier systems. They showed that, for Gaussian estimation error distributions, sum always performs better than vote, while for heavy-tail distributions vote could outperform sum. In the same year, Kuncheva et al. [Kun03b] defined upper and lower limits on the majority vote accuracy for individual classifiers. Again, they showed that a negative pair-wise dependency between classifiers is beneficial for all pairs of classifiers in the pool.

Interestingly, they also noted that diversity is not always beneficial. In 2005, Ruta and Gabrys [Rut05] proposed a revision of the classifier selection methodology, and evaluated the practical applicability of diversity measures in the context of combining classifiers by majority voting. In 2008, the book '*Model Selection and Model Averaging*' was published by Claeskens and Hjort [Cla08]. The book mainly explains, discusses and compares different model choice criteria, including Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Deviance Information Criterion (DIC), and Focused Information Criterion (FIC).

2.7.4. RANKING

In 1994, Ho, Hull and Srihari [HO94] introduced an ensemble of multiple classifier systems based on rankings for handwritten digit recognition. In 2009, Matwin and Kouznetsov applied the ranking method in order to boost precision and recall in a systematic review task [Kou09].

2.7.5. INCENTIVE METHODS

In this section, we discuss three effective works on ensemble learning that inspired the plan for our ensemble classification method.

In the year 2000, Jain, Duin and Mao [Jai00] introduced a list of combinations of classifiers. There are many reasons for combining multiple classifiers in the list (e.g. different feature sets, training sets, classification methods or training sessions), and the idea is that combining any composition of these might improve the overall classification accuracy. They had taxonomy in their experimental work. They trained twelve classifiers on six feature sets from a shared dataset, and used four methods of classifier combination,

median, product, nearest mean, and 1-NN (Nearest Neighborhood), across both the different feature sets and the different classifiers. Measuring performance against the best single result showed that: 1) there is no benefit from just combining different classifiers across the same feature set, and 2) there is substantial benefit from combining the results of one classifier across different feature sets.

In 2005, Fumera and Roli [Fum05] presented theoretical and experimental analysis of linear combiners for classifier fusion. The theoretical analysis showed how the performance of linear combiners depends on the performance of the individual classifiers, and the correlation between their outputs. In particular, they considered the improvements gained from using a weighted average, over the simple average combining rule.

In 2007, Hansen [Han07] showed that, since the error reduction could arise from reduced variance due to the averaging over many solutions, when individual members are selected the optimal trade-off between bias and variance should have relatively smaller bias, since the extra variance can be removed by averaging. If greater weight is given to the committee members that make better predictions, the error can be further reduced. The benefits of committee averaging are not limited to sum-of-squares error, but apply to any error function which is convex [Sew08].

3. Second Order Soft Co-Occurrence (SOSCO) Text representation

Text in natural language has a structure beyond the purely lexical level. Hence, it is reasonable to create a representation which can take into account the structure by modeling and processing the texts intelligently. This means that some type of first order (in logic terms) representation would be warranted. However, it is recognized that such representations based on first-order logic impose computational costs, which render them impractical for any kind of large problem. Thus, it is appropriate to seek representations which will achieve at least some of the expressivity of logic, in order to deal with structural data without incurring extreme consequential costs.

The utility of the second order representation of contexts was shown by Schütze [Sch98] and Pedersen [Ped06a], [Ped98a], [Pur04a] in many unsupervised schema. At this point we tailor our second order soft co-occurrence based representation method (SOSCO) for supervised learning tasks. In this chapter we introduce (SOSCO), which is intended to extract the inter-relationship between features (words) in a feature space, so they can be used for attribute filtering for the high dimensional space of the clustering process. This representation will be considered the basis for extracting the conceptual elements of the high-level context representation, which will be explained in the following chapters.

We begin by creating a word-word co-occurrence matrix (order-agnostic) over the entire corpus (where each row/column is a vector representation of the corresponding word) to represent each context (parts of a text that precede or follow a specific word or phrase). We simply extract corresponding vectors (row or column) for the words

contained in the context. We average the vectors word-by-word, and the average vector is known as the second order co-occurrence vector of the context.

In this chapter, we explain the second order weighted co-occurrence method in detail, as an augmented implementation of the second order co-occurrence method [Sch98, [Ped97a] that is specifically designed for *short text corpus representation*, including more than one context in each document, particularly for supervised text classification tasks.

In the rest of the chapter we explain the SOSCO representation generically, regardless of any specific application, and in the next chapter we will apply it in two practical case studies.

3.1. Preprocessing

In preprocessing, first all the headers (if there are any), internet addresses, email addresses and tags are filtered out, as well as extra delimiters like spaces, tabs, newline characters, and characters such as \ : () ` 1 2 3 4 5 6 7 8 9 0 \ = \ [] / < > { } | ~ @ # \$ \ % ^ & * _ + Expressive characters like “ - . , ; ‘ “ ” ’ ! ? ” are retained¹⁹, as punctuation can be useful for determining the scope of contexts. This step prevents us from including too many tokens as features in the text representations.

3.2. Closeness Matrix

We extract the term-to-term relations in a closeness matrix based on the Distributional Hypothesis, which states that words which occur in similar contexts tend to be similar. We then define a representation which takes into account the context (i.e. other words) and their mutual relationships, as acquired from the entire corpus.

¹⁹ (-) has been kept in order to individualize the compound words like: Multi-dimensions, Anti-oxidant, etc.

Assume a corpus contains n labelled²⁰ documents, $\mathcal{D} = \{d_i\}_{i \in \{1, \dots, n\}}$ consisting of vocabulary words v and $\mathcal{V} = \{w_j\}_{j \in \{1, \dots, v\}}$ where $\forall w \in \mathcal{V}; w$ is represented by the weighted vector $\vec{w} = \langle w_{w_x} \rangle_{x \in \{1, \dots, v\}}$, and $\langle w_{w_x} \rangle_{x \in \{1, \dots, v\}}$ is calculated based on a variety of co-occurrence pairs²¹ $(w, w_x)_{x \in \{1, \dots, v\}}$ in the document collection $\mathcal{D}$. Each w_{w_x} is assigned according to the different configurations²² of the co-occurring pair of words over the corpus $\mathcal{D}$. Consequently, each document $d \in \mathcal{D}$ is represented (at the sentence and document levels) by the aggregated vector $\vec{d} = \langle \mathcal{A}f(d, w_{w_j}) \rangle_{j \in \{1, \dots, v\}}$, where $\mathcal{A}f$ is an aggregate function (e.g. average, maximum, etc.) over the word vectors $\langle \vec{w}_c \rangle_{c \in \mathcal{C}}$, and $\mathcal{C}$ is the word sequence in the d (explained in detail in the next three sections), and creates the sentence representation of any sentence in the corpus.

3.3. Soft Co-Occurrence Matrix

After preprocessing, we begin to tokenize the corpus in order to build a soft co-occurrence matrix, in which the closeness of co-occurring pairs is recorded.

Definition of closeness: The closeness relation is determined by considering a variety of configurations of each pair of words in a sentence (our window). These configurations represent fixed relations between words, and are the tool we use to build a representation half-way between propositional and full-relational first order logic.

We want to observe that if we opted for a representation of any possible relation between words in First Order Logic (FOL), we would be working with a (logically) first

²⁰ Labels at this stage are not used and will be used in the higher representational levels in the next sections.

²¹ Will be listed and explained in the next section (3.3.)

²² The details are explained in Section 3.3.

order representation. We would certainly run into the expressiveness vs. efficiency trade-off; i.e. the FOL representation would be very expressive but also very inefficient and not scaleable. The vectorial BOW model, on the other hand, is a (logically) zero order (propositional) representation. We view our proposed representation, which considers relations, but from a fixed set, as a compromise between expressiveness and efficiency: we start with a predefined set of first-order configurations of relations between words, query the data for the occurrences of these configurations, and then work with the data representing the answers to those queries. This process corresponds to the propositionalization of the logical representation, and therefore we can position our representation half way between zero and first order logic.

The configurations of the word pairs inside the sentences are defined as follows:

- 1- Two adjacent words (bigrams, regardless of their order).
- 2- Two consecutive words with one word in between.
- 3- Two consecutive words with more than one word in between.
- 4- Two consecutive words with a comma “,” interval in between.
- 5- Two consecutive words with a semicolon “;” in between.
- 6- Two consecutive words with quotation marks “ “ ” in between.
- 7- Two consecutive words with “\r” or “\n” in between.

Note that we will never have pairs of words with any of [. ! ?] in between in a sentence.

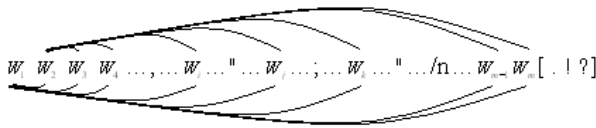


Figure 8. Illustrates the configurations of word pairs which can be extracted from a sentence (ending by either of the [. ! ?] symbols).

If condition #3 and #4 occur at the same time, we would consider it as configuration #4 of the co-occurrence.

Normally, co-occurrence is considered in a specific context, or in a window of limited size, such as 3 to 7 words before or after a target word, which would restrict the total context size from 7 to 15 words.

In this method, we select *sentence as our window unit* (see Figure 8).

Our method for extracting the closeness for semantic representation is based on a variety of word-word co-occurrence, according to the above definition of closeness. The algorithm uses a normalized integrated method to weight related terms in a document, based on word-word co-occurrence configurations in the entire corpus. The method considers the configuration of words inside each individual sentence in the corpus. Then, the document-word matrix is derived from word-word and word-sentence matrices.

If k is the number of tokens in a sentence, the total number of pairs extracted from a sentence can be calculated as:

$$\text{(Equation 1): } k + (k-1) + (k-2) + \dots + 1 = \frac{k(k+1)}{2}$$

This shows that the computational complexity of building the soft co-occurrence matrix M_C is in *quadratic in order of the typical sentence length*, which is less than 30 words ($30^2 = 900$), and *linear in the number of sentences* in a corpus.

We observed the latter linear complexity empirically, and it took a fraction of a second to process each short text.

Every individual configuration listed above applies an assigned coefficient factor (weight) for the accumulative closeness computation on each pair of words in the soft co-

occurrence matrix M_C . Furthermore, in order to minimize noise interference in the matrix, we simply decrease the coefficient weight of a co-occurrence when the assigned number in the above configurations list is increased. This means that, since the degree of relation between any two words in a co-occurred pair dramatically diminishes from configuration #1 to #7, the assigned weights should be decreased for configuration numbers 1 to 7. It also means the occurrence impact of those configurations on the matrix M_C , dramatically decreases from #1 to #7. The final weights are assigned by an empirical searching procedure.

We initially assign '1.0' to the closeness of each word X to itself. (X, X) and the values for the other regular pair of words (X, Y) in the closeness matrix M_C are calculated based on Dice's similarity measure for the closeness of any pair of words (X, Y) in a corpus, as follows:

$$(Equation\ 2): \quad M_{C_{X,Y}} = \frac{2(w_1 \cdot df_{1_{xy}} + w_2 \cdot df_{2_{xy}} + \dots + w_m \cdot df_{m_{xy}})}{df_x + df_y}$$

Where $df_x = df_{1_x} + df_{2_x} + \dots + df_{m_x}$ and C_{XY} is defined as the closeness of the pair of words (X,Y) , w_i ²³ is the assigned weight for the configuration number i , $df_{i_{xy}}$ is the frequency of co-occurrence of the pair (X,Y) in configuration i in the corpus, m is the number of distinct word pair configurations, df_x is the frequency of occurrence of the word X in the corpus, df_{i_x} is the frequency of occurrence of the word X in the

²³ We recall that the applied notation for "Words" was an upper case ' W '. Thus, the notation used for "Weight" in this section is a lower case ' w '. Based on a subset of the original Reuter-21578 dataset, the values (100, 35, 15, 10, 3, 1, 0) have been empirically chosen for the weights and used for all applications included in the thesis however those values can be fine-tuned for each application individually.

configuration number i with any word in the corpus, and df_y is the frequency of occurrence of the word Y in the corpus, and is calculated the same way as df_x . The values of C_{xy} in the matrix M_C are not normalized at this stage (due to applied W_i weights); they will be normalized after building up the matrix M_C ²⁴. In this matrix, cell values are indeed the measure of the closeness scores of pairs of words co-occurring in the sentences of the corpus. At this stage, approximately 75% of the values inside the matrix M_C are still zeros. Each row of the matrix M_C is a descriptive vector that represents the closeness of the features that co-occurred with the particular word indicated by the row name (see Table 1.).

	W1	...	Cop	Agriculture	...	W5000
W1	1.0000	...	0	0.0145	...	...
...	...	...	...	...	...	...
Robbery	...	...	0.9017	0	...	...
Trouble	...	...	0.7862	0.2012	...	...
Pesticide	...	...	0	0.7623	...	...
Apples	...	...	0.0002	0.3793	...	...
...	...	...	...	...	...	...
W5000	...	...	0	0	...	1.0000

Table 1. Sentence based Closeness Matrix (Symmetric) M_C ; an illustration of how closely words are interrelated

²⁴ We normalize all numeric values of each row in the matrix. The resulting values are in the range of [0,1] after normalization process.;The normalization process has been done by WEKA.

3.4. General and Domain Specific Stop Words Removal

Hans Peter Luhn is one of the pioneers of information retrieval. He coined the term “stop words”, and used the concept in his design. In text mining and NLP, stop words are words which are normally filtered out, before or after the processing of the original text corpus. The removal stage can potentially improve the level of discrimination in most natural language analysis/mining applications. These words could be listed by an expert human or by an automated procedure (see the following page). Although there is no explicit recognized list of definite stop words which could be applicable for all domains, in general cases we could point out some of the most common, short function words, such as “the”, “is”, “at”, “which” and “on”.

In our text representation method, if we remove the stop words from the text prior to determining the configuration of each pair it contains, there would be many changes among the configuration numbers (#1 to #3) and, consequently, to the corresponding effect on the co-occurrence matrix M_C . In other words, assuming that we remove the stop words in the first step, some words which actually have one or more words between them could be assigned to a configuration that is closer than the actual, and in this way the algorithm will overestimate the degree of co-occurrence. Hence, in order to perform stop word removal in the implemented system, we skip the calculation when one or both word(s) of the pair is/are in the stop list. Therefore, after building the soft co-occurrence matrix M_C as described above, we only need to remove the corresponding rows/columns

in which all the values are zeros²⁵ (this happens because we already skipped computations over the stop words). There are two groups of stop-words, which are removed in two steps: general stop words and domain specific stop words. First, we apply a general predefined stop word list that is appropriate for the domain we are working with (i.e. medical). Second, in some cases stop words are determined based on their frequency distribution, detected from the corpus after generating the word-word soft co-occurrence matrix M_C . We remove words with very high frequency relative to the corpus size and term distribution in both classes, since they do not help to discriminate between the classes. We also remove words/tokens that appear only once in the corpus, as they are usually not meaningful, and will not help the classification task since they appear with only one class, very possibly by chance.

$$\mathbf{W}_{\text{cop}} =$$

	Co
W_1	...
Robbery	0.9
Trouble	0.7
Police	0.8
Irrigation	0
Arrest	0.9
...	...
W_{5000}	

Table 2. Closeness Word Vector

²⁵ In addition to some individual words in our corpus (mostly names) which have not co-occurred with any other words in a sentence, there are some values in the matrix that remain zero because of co-occurrence with stop words that was skipped during the process.

As the word-word closeness is calculated regardless of word order in co-occurring pairs the matrix is symmetric²⁶, and the co-occurrences of any given word can be extracted from the corresponding row or column, as shown in Table 2.

3.5. Text Representation through Representation of the Containing Sentences

In this section, we describe a vector representation of documents in the corpus based on the feature space of the entire corpus (full dimensionality of the corpus) in two steps: Sentence Second Order Representation vectors and Document Representation vectors.

3.5.1. SENTENCE SECOND ORDER REPRESENTATION VECTORS

In the first step, in order to achieve the second order representation, each *sentence* of a *short text* in the corpus is represented by averaging²⁷ the contained features' vectors, which are extracted from the soft co-occurrence matrix M_C . The averaging process is normally applied to boost the role of features (we call them topical words) that co-occurred with the majority of the other words in a sentence (or a document). In this step, the soft co-occurrence matrix M_C does not include stop words; hence, the stop words cannot affect creation of the representation vectors. Therefore, we create the second order co-occurrence vectors by performing an averaging process among the containing words (their vectors) in a sentence. (Figure 9.)

²⁶ The symmetric attribute of the matrix is specifically helpful when we apply Singular Value Decomposition (SVD) in order to reduce dimensionality.

²⁷ The averaging function can be changed with another aggregation function, such as minimum (see Section 3.6. for details).

Sentence	I	Had	A	Good	Time.
	w_I	w_{had}	w_a	w_{great}	w_{time}
w_1	...	...	...	...	...
...	...	...	...	...	...
w_{5000}	...	...	...	...	...

$$\mu(\vec{s}_i) = \frac{\vec{w}_I + \vec{w}_{had} + \vec{w}_a + \vec{w}_{great} + \vec{w}_{time}}{5}$$

Number of words in the sentence.

Figure 9. Sentence Average Vectors (Second Order Representation) Input: Soft closeness vectors of all words in each sentence; Output: Sentence average vectors (Second Order Representation)

$$(Equation\ 3): \quad \mu(\vec{s}_i) = \frac{1}{|S_i|} \sum \vec{w} \in S_i ; \quad \vec{S} = \{\mathcal{A}\#(S, \mathbb{W}_{W_j})\}_{j \in \{1, \dots, v\}}$$

Where $\mu(\vec{s}_i)$ is the sentence representation vector of the i th sentence of the document, $|S_i|$ is the length of the sentence, and $\vec{w}$ is the vector representation of any containing word of the sentence S_i , as described in Section 3.4..

Since the words co-occurring with the words of any given sentence participate in the sentence representation vector ($\vec{\mu}$), at this stage the sentence representation ($\vec{\mu}$) has several times more non-zero features than the BOW representation of that sentence.

3.5.2. DOCUMENT REPRESENTATION VECTORS

In the next step, we calculate the document representation vector by averaging the second order representation vectors of the sentences inside the document. (Figure 10.)

We build the document representation from the sentence level (in two steps), rather than

directly from the word level; the sentences' second order representations have already been calculated in the first step.

$$(\text{Equation 4}): \mu(\vec{d}_i) = \frac{1}{|d_i|} \sum \vec{s} \in d_i; \quad \vec{d} = \{\mathcal{A}\#(d, \mathbb{w}_{W_j})\}_{j \in \{1, \dots, v\}}$$

Document		
	It was Sunday.	S1 (SOSCO vector)
	I played with the dog.	S2(SOSCO vector)
	We had a great time.	S3 (SOSCO vector)
	...	...
	And I woke up.	Sq (SOSCO vector)

Figure 10. Document Representation Vector

where $\mu(\vec{d}_i)$ is the document representation vector of the i th document of the corpus, $|d_i|$ is the number of sentences in the document, and the $\vec{s}$ is vector representation of any containing sentences of the document d_i . Performing the document level aggregation (averaging) is another step toward increasing the number of non-zero elements of the text representation vector (Figure 11.)

Regarding the above detailed explanations, we define the *SOSCO representation* as a second order co-occurrence based text representation method in which the word-word corpus range relation is calculated according to (Equation 2) at page 64.

Computational complexity for the SOSCO representation can be estimated based on Equation 1 to 4 as follow:

$$O(K^2) + O(n) + O(n) + O(n) = O(n)$$

In which K , is the average number of words inside a sentence of the corpus (*the typical sentence length*) that has been shown that can be considered as a constant value;, and n is the number of words of the corpus.

The performance of the second order representation of short texts already has been studied and discussed for non-supervised learning purposes by Pedersen [Ped05a], [Ped06b], [Ped07].

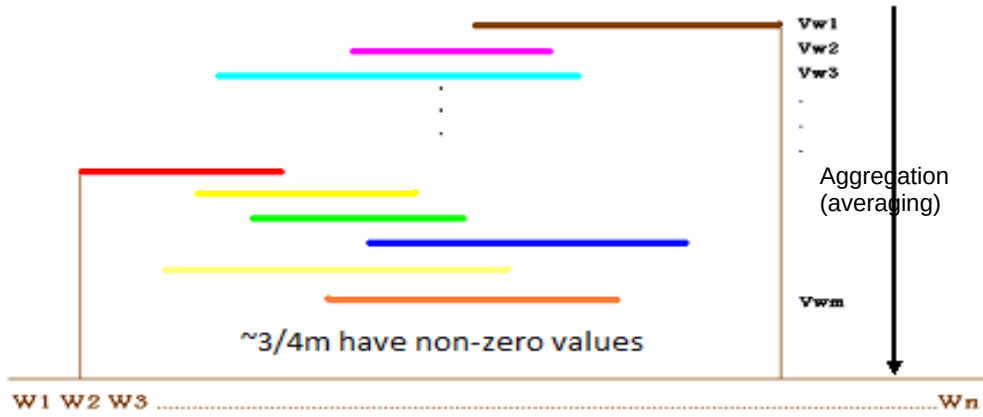


Figure 11. This graph shows how the SOSCO representation of a sentence in the corpus contains about 3 quarters of m , the corpus feature space attributes with non-zero values²⁸; $V_{wi} = \vec{w}_i$. The actual presentation of each line is a sequence of dots with some gaps in between, which for better illustration and simplicity has been depicted as a solid line. Axis X is the feature space (words) while the axis Y is about all the representation vectors that participate in the aggregation process.



Figure 12. This graph shows how the SOSCO representation of a document in the corpus contains about 90% of the corpus feature space attributes with non-zero values; $V_{si} = \vec{s}_i$. The actual presentation of each line is a sequence of dots with some gaps in between, which for better illustration and simplicity has been depicted as a solid line. Axis X is the feature space (words) while the axis Y is about all the representation vectors that participate in the aggregation process.

²⁸ The numbers in Figures 10 and 11 are based on our experiment results for the applications listed in the next chapter.

After two level of vector aggregation (averaging) almost 90% of the initial features over the entire corpus feature space are non-zero at this stage (Figure 12.).

Although the value of any cell in the vector is an indicator of the associated power of the corresponding feature in the vector space with the sentence/document that contains it, this value does not directly indicate if the feature occurred in the sentence/document or not; it globally represents the relevance level of the sentence for each dimension (each feature). Using the algorithm for creating our text (document) representation vectors, we implicitly imposed a certain type of smoothness in the feature space, compared with the first order (i.e. BOW) representation of the same text. In other words, in addition to computing the explicit participation of a given feature in a given document, we compiled the participation of other related features with respect to each one's closeness to the given feature. This means that, even if we eliminate one of the features from the feature space (after creating the soft co-occurrence matrix M_C), we can still expect to retain the discriminatory power of the classification task, if that feature co-occurred sufficiently with some other related features in the corpus. (This assumption can usually be relaxed for adequately large corpora.)

Considering the above, we can perform a proper feature selection algorithm with the expectation of preserving enough informative power for the next level, the classification task. Furthermore, after feature selection (which preserves a subset of features with lower levels of dependencies), in addition to common classifiers for text²⁹ we can also use further classifiers, which cannot perform well or are fast enough over high dimensional

²⁹ Classifiers which can handle numerous features.

data. This means using 200 to 1000 features for building more discriminative classification models in even less processing time.

3.6. Contrast Parameter

Upon reviewing a variety of text classification projects, we find a number of conceptually different tasks. Sometimes the task is to classify texts into topic classes, such as medicine, agriculture, economy and so on. Other projects classify texts based on some restricted predefined conceptual domains, or even based on sentiments or emotions that might have been expressed by the writers. Obviously, for topic identification there are some distinct keywords which play an essential role in the classification task, but in other cases, such as sentiment or emotion analysis, we cannot rely on these keywords alone.

With regard to the steps for creating our text representation vectors, since corresponding dimensions of words which have co-occurrence relations with the occurring words in a document have non-zero values, we observed smoothness in the vector representation, compared with strong contrast in the BOW representation of the same text. In other words, in a BOW vector representation of a text, a value is zero if the word doesn't appear in the text; or if it has some value which means the word explicitly participated in the text, and there is nothing in between. However, in the SOSCO representation, a feature can have a non-zero value based only on its closeness with the words in a text, and not explicitly appear in the text. This characteristic of the SOSCO representation causes a smoothness among the related feature values (in the representation vector), compared to the extreme contrast in the BOW representation vectors of the same text.

Comparing the results of the two representations on several datasets/applications, we observed the advantages of each in different domains. Hence, we decided to define a contrast parameter on the SOSCO representation, based on a range of applied aggregation functions in the algorithm of the SOSCO. For each sentence in the stage of sentence representation vector, we run the algorithm not only based on the *average* aggregate function, but also on the *minimum* and *maximum* aggregates (instead of only *average* in generic format). Thus, for each dimension of the sentence representation vector, there is a range of discrete values of the minimum (values of that dimension in the containing word vectors of the sentence), the average and the maximum. We assigned a range of +9, 0 and -9 contrast values to each of these discrete values, to allow calculation of other values corresponding to the other contrast values in the range (between -9 to +9; e.g., +4).

Therefore *the contrast parameter* indeed is a parameter that can be tuned for a variety of text representation tasks in which for any $l = 0$ to $+9$ ³⁰ as contrast parameter value, values of the sentence representation vectors are calculated as:

$$\{\text{Avg}(S_i, \mathbb{W}_{W_j}) - l * (\text{Avg}(S_i, \mathbb{W}_{W_j}) - \text{Min}(S_i, \mathbb{W}_{W_j}))/9\}_{j \in \{1, \dots, v\}}$$

Hence, the contrast parameter allows the value of each feature (in the feature space) to vary between the average and the extremum (minimum and maximum) values of each feature among the word vectors of a sentence inside a document. This means that, based on the defined range of 0 up to +/-9 for the contrast parameter, we will have a range of values for each dimension of the SOSCO representation vector for a given sentence. For the two highest points of contrast (levels +/- 9), the averaged second order soft co-

³⁰ Values for any $l = -9$ to 0 also can be calculated in a very similar formula, however we do not suggest applying values less than 0 for most of the text classification tasks.

occurrence vectors lie in the center level of smoothness³¹ (contrast level 0). We set the maximum value to 9 empirically, in order to have a limited number of values available to try to find the optimum for each application over a held out subset of documents (finding the best value of the contrast parameter for any task can be considered an optimization problem). Obviously, there is an optimum value for the contrast parameter, which gives us the most discriminative power for machine learning algorithms for each application. We ran many experiments on a variety of applications and input data, with different targeted conceptual levels, and we empirically observed that the optimum contrast value for the topic/subject identification tasks is higher than that of some sentiment/emotion analysis tasks. However, it can be interpreted by the extensive role of certain keywords in the topic/subject identification tasks.

We also observed that applying different values for the contrast parameter can result in different aspects of the represented text. Hence, the contrast parameter cannot only be used for finding the best fit representation of the text in any given application, but can also be applied to obtain a variety of representations of the same text for a committee of ensemble learners.

However, while getting the benefit from the both of high contrast (+9) and low contrast (0) representations at the same time, practically is possible we always prefer to use both instead of one of them; or a representation with a contrast parameter in between. For this reason in the rest of this thesis we almost in all cases use a contrast equal zero and

³¹ Note that both the BOW and second order co-occurrence representation vectors are already normalized (contain values between 0 and 1).

simultaneously also take the BOW high contrast representation into account to improve the classification performance.

3.7. An Example

To illustrate the methodology so far with an example, we start with a short text:

*“The **Congo River** (also known as the **Zaire River**) is the largest river in the heart of Africa. Its overall length of 4,700 kilometers (2,920 miles) makes it the second longest in Africa (after the Nile).”*

First, we recall the existing symmetric word-word soft co-occurrence base closeness matrix M_C , which was created based on the entire domain (corpus) $d \in D$, as shown at the end of Section 3.3. In this matrix, each row (column) represents each word of the corpus, according to its different co-occurrence configurations (based on the weights) within all the sentences in corpus. Hence, assuming an extensive domain,³² the sample word “heart” could potentially be co-occurring with terms like cardiovascular, attack, cholesterol, blood-pressure, surgery, etc. (G1)

In some other contexts, “heart” could be co-occurring with another group of terms, such as sweet, broken, love, like, dear, feeling, darling, deep, cordially, pureness and so on. (G2)

However, the word “heart” could be co-occurring in some contexts talking about some geographical locations, like the names of some cities, countries, territories, states, or other smaller or bigger locations with the meaning of “Center”. (G3)

³² Otherwise the level of ambiguity will be lower, and the representation would be fairly straight forward.

Finally, coincidentally, “heart” could be co-occurring with some other scattered words with no clear relation to each other (Figure 13.).

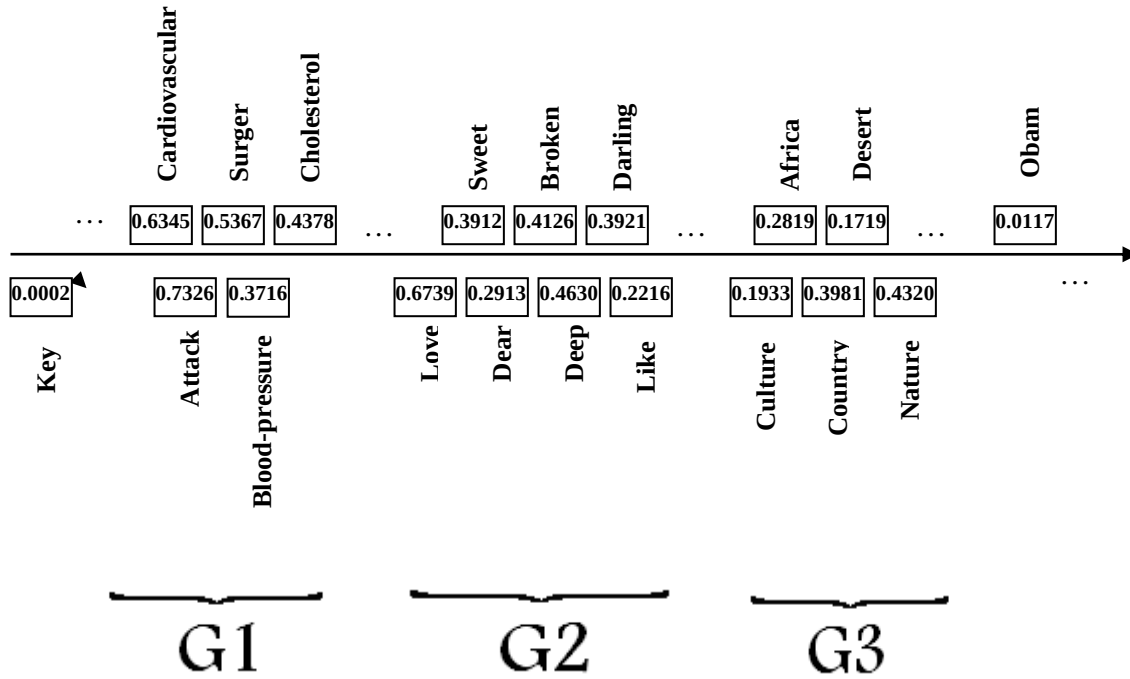


Figure 13. The above figure illustrates the vector representation of the word “Heart” as an example over the entire domain of the task (corpus).

When we build the sentence level representation vector for our first sentence (see Section 3.5.1.): “*The **Congo River** (also known as the **Zaire River**) is the largest river in the heart of Africa.*”, the vector representation of each word of the sentence (including “Heart”) is going to be totaled to create the vector representation of the entire sentence (e.g. Congo, river, also,..., Africa in Figure 14. and 15.).

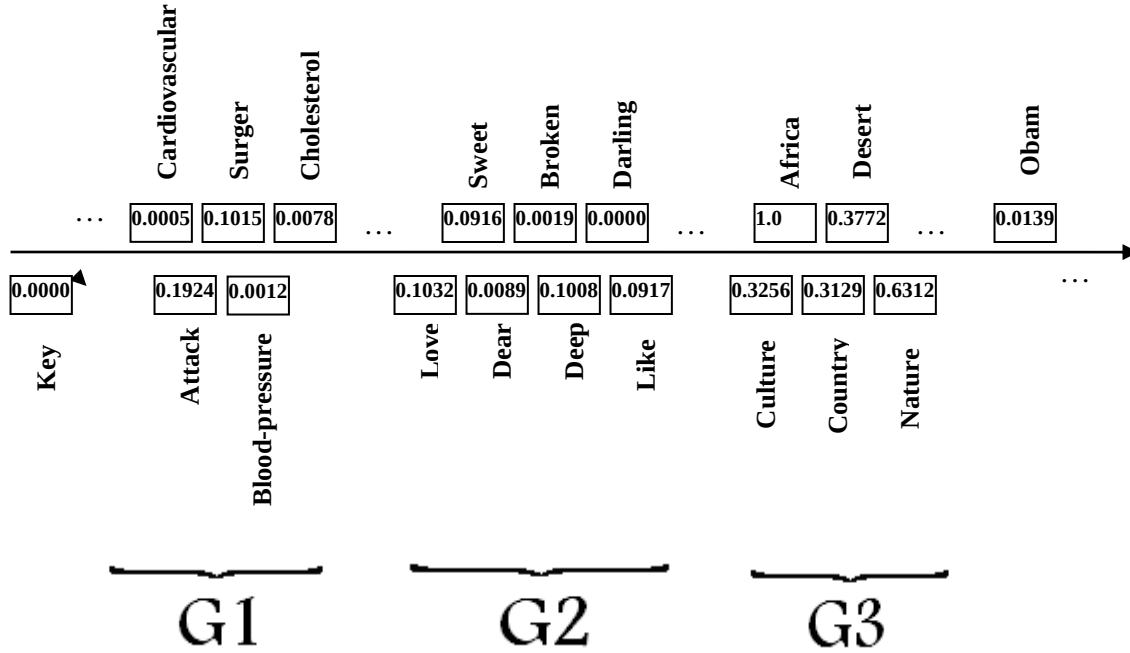


Figure 14. Example of illustrative vector representation of the other word “Africa” over the entire corpus.

While aggregating the word vectors, every attribute of the domain vocabulary is averaged to build up the sentence representation vector. The averaging process causes the values of the irrelevant attributes in G1 and G2 of the “Heart” vector to be dramatically diminished—by adding to the vectors of the other word in the sentence with practically no significant values in G1 and G2, then dividing by the number of words in the sentence—and the values of the relevant attributes in G3 clearly dominate (Figure 14.).

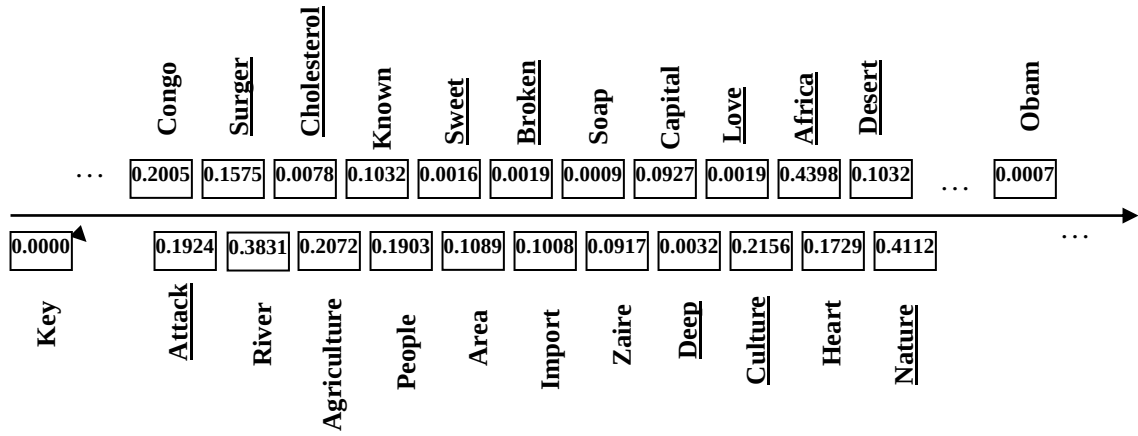


Figure 15. Illustrative vector representation of the first sentence: “The Congo River (also known as the Zaire River) is the largest river in the heart of Africa.” over the entire corpus. Values of the underlined attributes (from G1, G2 and G3) are well tuned (i.e. are considerable if in G3, and diminished otherwise) after the aggregation process.

We then repeat the same process for the other sentence in the text, after which we will have two individual average vectors which can be aggregated in the final stage into one vector (in the document level).

3.8. Advantages of the SOSCO approach

The basic co-occurrence method and its descendants have mainly targeted word sense disambiguation and topic detection tasks. Those were generally applied to unsupervised clustering tasks. Hence, it is not easy to compare them with the current method, which is designed to be applied to supervised learning.

We believe the following are the contributions specific to our method:

- The capacity of the SOSCO representation of a text includes more than the local context, and may be interpreted based on the entire corpus.

- SOSCO could be applied for the topic representation tasks since during the stages of aggregations (averaging) we automatically boosted up the role of topical words.
- Since any bigram in the corpus already has a closeness value in the co-occurrence matrix M_C , moving a window of size two (bigram) over the text of any document $d \in D$ (from the beginning to the end, in order) creates a continuous closeness graph along the text (Figure 16.) We call this Text Progression Representation (TPR)³³.

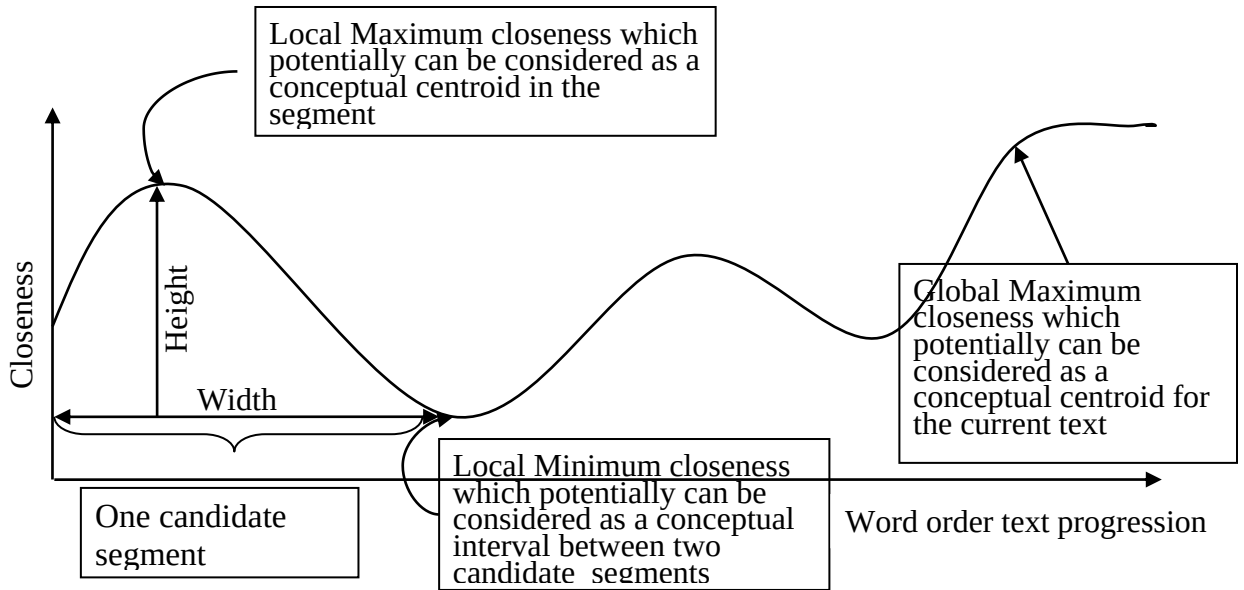


Figure 16. Word order Text Progression Representation (closeness based) graph. Plot shows the closeness between the vectors for the bigrams in position n and the vector for the position $n+1$ (for example) in a given document.

TPR is recorded as a sequence of normalized numbers which can be considered in *segment detection* (text segmentation) [Cai06], as well as a characteristic (fingerprint) for the documents (with consideration of its words in order). This sequence (TPR) can be created and tagged to the documents in linear time order of the length of the documents

³³ TPR reaches a maturity level when it is polarized to its essential conceptual elements (in Section 9.2.) as partial Text Progression Representation (pTPR).

(~350 words in average) which is quite fast, and can also be used for other purposes like summarization and identification. (See more in section 9.2.)

- Polysemy is another problematic phenomenon for knowledge extraction and information retrieval systems. Polysemy describes words with more than one sense/meaning, which is a common property of any language. Polysemous words can potentially be considered ambiguities that significantly decrease the quality of knowledge extraction. SOSCO automatically degrades the untargeted sense/meanings of words, while boosting the targeted sense/meaning of the word in the sentence/document by calculating (averaging) the first order co-occurrence word representation of the words in the sentence/document level. For example, assume the word “bank” in a sentence is pointing to the word “bank” as in river bank, and there are enough other supporting words around the “bank” to make it recognizable to humans.

The values of the features (in the sentence or document vector) which exclusively have closeness with the other meanings/senses of the “bank” would be suppressed while the SOSCO is running the aggregate function (i.e. average). This is due to the dissimilarity of the rest of the words in the context with the other meanings/senses of “bank”. Hence, the attributes which are related and express the right meaning of ‘bank’ would emerge in the sentence/document representation vector.

- It handles feature sparseness robustly, with only ~10% zero values in the feature space³⁴.

³⁴ The SOSCO method represents texts based on co-occurrence of features in the whole corpus, rather than an individual phrase, sentence, segment, document or topic.

- Soft co-occurrence is one of the most important advantages of this method. SOSCO, unlike the other co-occurrence based methods, applies different weights based on different word-word co-occurrence configurations, which makes it a relational associative approach rather than a solely associative approach.
- Although most of the popular vector space representation models essentially relax the assumption of word dependencies, there are actually strong associations among the words in any language. The interesting point is that, although the word independence assumption is reasonable in the first-order representation, it is possible to get improved performance by using word associations in a variety of information retrieval or knowledge extraction problems, such as topic search through a corpus. For example synonymy is usually problematic in systems which represent documents based on their words. These kinds of retrieval systems have trouble discovering documents with the same topic that employ a different vocabulary. When we apply second order co-occurrence in sentence and document levels, it means documents can be evaluated as similar even when they have differing vocabularies.
- As we apply the second order co-occurrence for building our matrix M_C , there is inter-dependency between the features of any representation vector calculated in the matrix M_C . This means that the value of each element in the representation vector not only *carries some information about the exact corresponding word*, but also *includes something about some closely related words* in the feature space (e.g. synonyms), which means that each element carries some load on behalf of its closely related features³⁵.

³⁵ This could be imagined by similarity to the RAID technology for restoring the lost data or estimating one variable by factor analysis in a system.

Therefore, by performing the feature selection process, not only do we retain all the information about the removed features, but we can also expect to have enough information to achieve promising performance for a variety of machine learning algorithms which could only be run on limited number of features (See table 6.). The ability to bypass the LSI [Dee90] dimension reduction procedure, which is usually the most computational- and time-consuming step in similar tasks, due to using the text vectors with less than 10% sparsity. Experiments show (first row of table 6.) that these vectors are carrying enough information that, after applying a simple feature selection step, we would still have enough information for the classification task.

- Every aspect of the SOSCO representation is designed to extract the word associations [Dee90] which do not impose an unsustainable (quadratic) computational complexity.
- SOSCO increases the representation power by taking the information granularity of sentences into account (this can be modified for other types of segmentation), and building text representation vectors on two levels (sentence level and text level), rather than one.
- The reduced feature space obtained after the above process is far more human-understandable than the LSI final feature space obtained after performing the Singular Value Decomposition (SVD) step.
- The serializability of all the above processes has been proven empirically. These processes can be executed sequentially with linear complexity on a number of short texts.

3.9. Limitations

- The SOSCO text representation method contradicts the independence assumption in some machine learning algorithms (e.g. Naïve Bayes). However, this limit can

be relaxed by applying feature reduction algorithms in which the ultimate selected features have less dependency. This problem will be addressed by TOR-FUSE algorithm in Chapters 5 to 7.

- Finding an appropriate contrast parameter value sometimes requires spending considerable time in the development step.
- We do not suggest the SOSCO representation for long-textual corpora since the internal two level vector aggregation process (e.g. averaging) will not come to an informative and expressive representation vector for long documents.

In the next chapter we present two empirical applications of the SOSCO representation.

4. SOSCO in Practice

In order to test the second order soft co-occurrence (SOSCO) representation on a variety of short text corpora,³⁶ and to perform preliminary modifications for improving the representation power in the detail level (fine information granularity), we applied the method on the following two text analysis tasks.

In the first, we describe a project undertaken by an inter-disciplinary team of researchers in sleep and machine learning. The goal is sentiment extraction from a corpus containing short textual descriptions of dreams. Dreams are categorized in a four-level scale of affects.

4.1. Classification of Emotional Tone of Dreams

Research in psychology shows that emotion is a prominent feature of dreams [Dom03], [Hob98], [Ong05]. Typically, the level of emotions or sentiments in dreams is assessed by content analysis made by human judges using scales with various levels, or by the dreamers themselves. Most of the studies on dreams have used time-consuming coding systems that depend on a rater's judgment. Hence, it is practical to develop an efficient means of scoring dreams that can be used with large data banks and reproduced across laboratories. Exploration of dreams' emotional content using automatic analysis has been defined. A sample of 776 dreams, reported in writing by 274 individuals of varied age and gender, was used for word-correlation analysis.

A subset of 477 texts was rated by a judge using two 0–3 scales, which indicated the negative or the positive orientation of the text describing the dream.

³⁶ The method has also been applied on languages other than English.

A voting committee of different classifiers provided the most accurate results, with the least mean squared error [Raz09].

The agreement between machine rating and the human judge scores on a scale of 0–3 was 64% (Mean Squared Error 0.3617). This was 14% more than previous results on the same task, which was based only on the BOW representation method [Nad06]. It was also significantly better than the chance probability of 25% and a baseline accuracy of 33% (percentage of the most occurring class). The results demonstrate that estimates were, at most, one level away from the human judge score [Ibid], and indicate promise for the automatic analysis of dream emotions, which is recognized as a primary dimension of dream construction.

Many works (e.g., Turney [Tur02]) found similar problems of classifying texts as positive or negative, usually as a binary classification. This formulation differs from our 4-level scale, which is motivated by the need for fine-grained analysis of sentiment strength for further processing (e.g. analyzing the stress levels of dreamers). We consider our problem formulation more difficult than the binary classification, but it is also more flexible.

The granularity of our scale (4 levels) was chosen to reflect the variety of sentiment experience, and to maintain simplicity. One application of this measurement that we envision is assessment of the stress experienced by the dreamer. Previous work aimed at drawing a link between negative sentiments in dreams and dreamer stress relied on content analysis of written dreams [Del02b]. The negative scale we present can be used as a feature in a larger system for stress analysis. A more general application of automatically analyzing dream sentiments would be the mining of large dream data

banks, and the discovery of unexpected data about sentiments in the dreams of individuals of different age, social status, etc. Hence, from a machine learning perspective, the task of dream sentiment analysis is expressed as a classification problem, with labels $\{0, 1, 2, 3\}$. The goal of this work is to create a system that can reliably replace a human in analyzing sentiments in dreams. In the next sections: the related works are discussed, the dream corpus is explained, our methodologies to extract the appropriate features and the classification task are detailed and, finally, our experiments and results in automatic dream sentiment analysis are presented and discussed.

Dreams were obtained from a dream bank created during a normative study conducted at the Sleep Research Laboratory of the University of Ottawa.

A previous study on a very similar task in the psychology department showed [Nad06] that the inter-judges agreement varied between 57.7% and 80.8%. This agreement was low for the positive scale compared to the negative scale, and the score on the positive scale was not well differentiated from one dream to another. Works in dream analysis often concentrate on negative sentiments, since they are typically more common and differentiated than the positive sentiments [Hal66], [Har98]. Therefore, as the negative scale can be useful in isolation, we excluded it for subsequent analysis.

The Linguistic Inquiry and Word Count (LIWC) dictionary [Pen01] provides measures of the percentage of positive and negative words in texts. The dictionary is composed of 4500 words and word stems. This resource makes no use of disambiguation rules; it relies on simple word counts. The value of LIWC is its scrupulous choice of words made by multiple experts that came to near perfect agreement.

Hence, we decided to use the LIWC dictionary to identify affective words, and the CMU Link Grammar Parser to identify adverbs. The reported LIWC affect was modified based on different types of modifiers, like negations and modalities handling (*e.g. very, extremely, etc.*), for better representation. This allows indication of when the context changes the polarity of a word (*e.g. the passage “is not kind” means the opposite of benevolent, charitable*). We also attempted to develop a novel dynamic representation of changes in affect with respect to dream progression.

4.1.1. DREAM ANALYSIS IN PSYCHOLOGY

Sentiment analysis is an important component in the study of dreams, since emotions are considered largely responsible for structuring the content of dreams [Har98], [Nie05]. Recent findings from brain imaging studies have shown increased activation of the limbic and paralimbic areas during Rapid-Eye Movement (REM) sleep [Maq96]. Because dreams are strongly associated with REM sleep, this may account for the emotional intensity of dreams [Dom03]. However, further study is needed to better understand the origin and potential role of emotionality in dreams. Up to now, most recent studies on dreams use the classical scales of Hall and Van de Castle [Ong05], which are considered the most detailed and complete coding system available for scoring dreams [Dom03]. The system is comprised of various scales which measure both positive and negative content, such as the presence of friendly or aggressive interactions, emotions, good fortunes or misfortunes, and successes or failures. However, this system is time consuming, and depends on the ranker's judgment. It is very important to develop an objective method of scoring dreams that is independent of human judgment, and can be reproduced across laboratories. So far, automatic analysis has not been used in studies of

emotions in dreams. The development of this technology could improve our knowledge of dreams, and potentially provide a major breakthrough in this research area.

4.1.2. DREAM BANK

Dreams were obtained from a dream bank created during a normative study conducted at the Sleep Research Laboratory of the University of Ottawa. The ethics committee of the university approved this normative study, as well as the use of the dream bank for future studies. Volunteers were informed that their dreams could be used in other dream studies, and they gave their consent. Their participation mainly consisted of completing a brief dream diary at home during a maximum period of three weeks, and writing down all the dreams they remembered when they woke up, up to a maximum of four dreams. The dreamers were asked to subjectively rank (0-3) each of the following 10 features of their dreams: "Joy, Happiness, Apprehension, Anger, Sadness, Confusion, Fear, Anxiety, Negative Affect and Positive Affect". Among those 10 features eight are emotions, and the last two give an overall subjective emotional valuation of the whole dream.

A sample of 776 dreams, reported in writing by 274 individuals of varied age and gender, was used in the study. A subset of 477 texts was rated by a judge using two 0–3 scales describing the negative or positive orientation of the dream content.

4.1.3. SAMPLE DREAM

"I was in an archaeology class. We were out in the field looking at a grave site. We were passing around lots of pottery from the site. Our teacher told us that truffles grow where the roots of regular mushrooms grow into soil that contains decaying flesh-like a grave site. I knew the value of truffles so I stuck my hand into the loose side of the grave and

fished out several truffles. Later I was at my own excavation site. It was a large tomb with a giant sarcophagus. Inside was a large bed with a dead dog in it. But the dog wasn't mummified. It was just rotting. There were bugs and maggots and beetles everywhere. As I moved the dog and the pillows and sheets, more maggots and beetles fell out. Soon the sarcophagus and bed and floor were covered in creepy crawlers. Luckily I'm not squeamish or afraid of bugs. But then a giant spider dropped from a sheet I had just pushed aside. I screamed and ran from the sarcophagus, down a hallway, but the spider followed me and was soon joined by another similar spider. I was very scared....”

4.1.4. METHODOLOGY

In order to build a representation adequate for the classification of dreams with respect to their affective content, we have decided to exploit three kinds of information describing the dream from the emotional perspective: the semantics of the natural language description of the dream, the dynamic affective change in the dream, as well as the self-assessed feelings of the dreamer about her dream. This led us to a 3-partite representation in which the semantic part was built directly from the text describing the dream using lexical properties of the whole corpus, the dynamic one was using NLP techniques as well as specialized dictionaries, and the subjective one was taken directly from the data. We have selected from each representation the features most important for classification, and then performed final training of the classifier on their union. Below we describe each of the three parts of the representation and the final combined representation of dream descriptions. To introduce the applied methodology, we mainly employ the specific representation method with which we could target the most appropriate conceptual level

of the dream texts, then show how to apply it to obtain the most powerful sentiment analysis from it.

As explained earlier, a sample of 776 dreams, reported by 274 individuals of varied age and gender, was chosen for the dream sentiment analysis task. From these, a pure English subset of 477 tagged dream descriptions was used for training and testing the classification models (by 10 fold cross validation). The dreams were rated by a judge and by dreamers, using two 0-3 scales as described.

For the proposed method, we first create the word-word co-occurrence matrix M_C over the whole corpus of individually labeled dreams (each row/column is a vector representation of the corresponding word), then, to represent any context, we simply extract and average the corresponding vectors for the words contained in the context. We recall averaging the vectors word by word, to obtain the second order co-occurrence vector of the context.

We use the Second Order Soft Co-Occurrence (SOSCO) method with contrast value equal “0” which is equivalent with the normal mean aggregate function; the method is designed for short text corpus representation (including more than one context in each entry), particularly for supervised text classification.

4.1.4.1. Dynamic Representation

We recall that we selected the LIWC [Pen01] dictionary to assign the initial positive/negative affect to every word in a dream. Hence, after applying the LIWC dictionary on the dream corpus, any word found in the dictionary has an affective tag, which is initially the one extracted from the dictionary (+/-1). Next, we used the CMU Link Grammar Parser to identify adverbs. The parser helps us to identify some affective

modifiers (*e.g. very, too, extremely, etc.*). In this step, the initial affective tags are modified according to the extremity level of the modifier, which can be found in a modifier table in the system (Table 3.). The values in the table are assigned based on the extremity level of the modifier, and adjusted by running many iterative experiments for adaptive learning based on a separated dataset including 100 dream diaries used in the previous study [Nad06]. In the next step, we further modified the assigned affect values if they were under the influence of negations such as not, non, un-, im- and so on. We noticed empirically that, contrary to what we first expected, when a negation modifies any type of adjective/adverb, its affective influence on the containing context is not completely reversed. For example, if we find an affective word like “happy” in a sentence with positive affective value (+1), the tag (-1) for “not happy” is not the most appropriate one; or when we have “so happy” with positive affective value of (+3), for “not so happy” the value (-3) is too low, as this value is normally assigned to an expression like “too sad”. In other words, when we say “He is not happy!” it does not mean that he is sad, but that we expect to see him mostly happy and he is not. Also, when we say “He is not so happy!” it definitely does not mean “He is too sad!”. Hence, we empirically created our lookup modifier table, as illustrated in Table 3.

Modifier	Value
Extremely,...	Add 3 (Same sign)
Too, So,...	Add 2 (Same sign)
Very,...	Add 1 (Same sign)
Not,...	Add 1 (Opposite sign)

Table 3. An example portion of the modifier table in the system. Each value is added to the initial affect value extracted from LIWC, considering the initial affect sign and the above corresponding value sign.

These two-step modifications allow us to recognize if the context polarity changes for the next steps of our process (Table 4.).

	LIWC		LIWC'	
Dream	Pos	Neg	Pos'	Neg'
It ends at the brink of a steep hill very grassy and green not at all <u>threatening</u> (-1).		1		0
I was <u>too scared</u> (-1-2) to watch it so I went in the kitchen where my family was eating.		1		3
We had a big family reunion in a huge <u>party</u> (1) room. I was so (+1+2) <u>happy</u> .	2		4	

Table 4. An example of the applied affection modification. The first two columns are based only on the LIWC dictionary, and the last two columns show the weights after modifications.

In the next step, we applied the final affective tags in order to explore the affection tendencies and variations through the progress of the contextualization of dreams. (Figure 17.) Since such graphs provide a useful visualization of the dynamic emotional progress of a dream, we propose to call them “onirograms”.

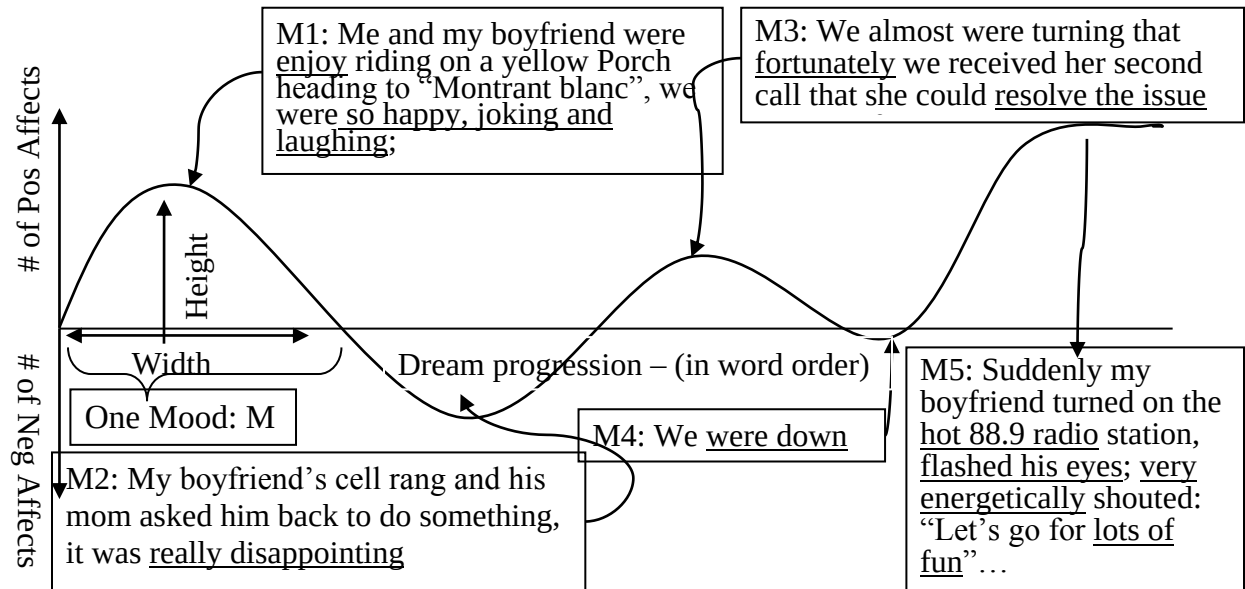


Figure 17. Affection Polarograph (onirogram): An illustration of the polarity and emotional tone of the contextualization of dreams over the time.

To accomplish this, we extracted a variety of new sentimental attributes based on their progression along the dream text, in order to boost the discriminative power of our contextual classification task (see Table 5). For example in the sample dream presented in figure 2, the first mood M1 has the height of 5 in positive affect and width of 22 words before stop word removal and 12 after removal. The initial mood of the dream is positive and the number of positive to negative mood shifts is 2. The number of positive moods in the dream is 3 while the number of negative moods is 2, etc.

Attributes directly from the onirogram (direct attributes)	Attributes calculated from the direct attributes			
Height positive affect for any positive mood per dream	Max	Min	Average	St.Dev.
Width positive affect for any positive mood per dream (number of words per mood)	Max	Min	Average	St.Dev.
Height negative affect for any negative mood per dream	Max	Min	Average	St.Dev.
Width negative affect for any negative mood per dream (number of words per mood)	Max	Min	Average	St.Dev.
Initial mood affect (Pos. or Neg.)	Width		Height	
Average dream affect	Pos. or Neg.		Value	
Number of Pos. moods	Number of Neg. to Pos. Variations		Number of Pos. to Neg. Variations	
Number of Neg. moods				
Total positive affect (before modification)	Normalized (divided by number of words per dream)			
Total negative affect (before modification)	Normalized (divided by number of words per dream)			
Total affect(before modification)	Normalized (divided by number of words per dream)			

Width negative affect for entire moods per dream (number of words)	Normalized (divided by number of words per dream)
Width positive affect for entire moods per dream (number of words)	Normalized (divided by number of words per dream)
Total moods per dream	

Table 5. The attributes extracted from the onirogram (directly or indirectly)

4.1.4.2. Attribute Selection

We have experimented with different levels of feature selection, and we have found out that the aggressive election shown in the following table gives better results, compared to less aggressive selections. Following the feature selection, the three introduced representations are combined into a single vector, which becomes the training set for the machine learning classifier. The labels in the training set are the labels given by the human judges. We applied the Relief Attribute Evaluator [Mar07], using the Ranker selection method from the Weka machine learning toolkit (Table 6.) [Wit05].

Attributes Groups	# initial attributes	# attributes after attribute selection	% of category after attribute selection
Text	4618	39	0.9%
Progression	36	21	58.3%
Demographics	2	2	100%
Dreamer Emotion	8	8	100%

Table 6. Attribute selection results

4.1.5. CLASSIFICATION RESULTS

We selected two evaluation measures for our experiments. First, we calculate classifiers' accuracy, the sum of correct guesses over the total number of guesses i.e., their performance at exactly finding the right label (e.g., human rates 3, machine guesses 3

would be a correct guess). Second, we calculate the mean squared error of classifier the average of the squares of the differences between the human labels and the machine predictions. This metric is low when a classifier guesses close to the human value (e.g., human rates 3, machine guesses 2) and becomes high if the classifier is far from human judgment (e.g., human rates 3, machine guesses 0). We report results for stratified 10-fold cross-validations. The baseline accuracy is given by a classifier that always guesses the majority class. In our dataset, 30% of the dreams were rated with label “2”; this is the majority class. Therefore, always guessing “2” results in 30% baseline accuracy. After performing feature selection, we ran many simple and ensemble learner algorithms on a variety of compositions of selected attributes.

In the next step, we review the Second order co-occurrence representation of the contextual content of the dreams, and start by performing feature selection. We then ran multiple simple and ensemble learner algorithms on a variety of compositions of selected attributes, applying 10 fold cross-validation. In this step, if we compare our accuracy using the Second order co-occurrence representation method (55%) and the accuracy of the previous work [Har98] (38%).

At the next level, we combined all the selected attributes to determine the most discriminative classifier, in order to achieve the highest agreement with our psychologist labels. A voting committee of three Adaboost and two Bagging meta-classifiers³⁷ provided the most accurate results with the least mean squared error on the prediction of negative affection, with an accuracy of 64%; significantly better than the baseline

³⁷ The simple classifiers used for the above classifiers were Multinomial logistic regression and J48 decision trees.

accuracy (30%) and the chance probability (25%). The mean-squared error was 0.3617, which means almost all errors had a difference of only 1 on the scale. With these results, we could predict 13% more efficiently than the previous work on the same task, which was based only on the BOW representation method.

The results indicate that estimates were, at most, one level away from a human judge score³⁸, and offer a promising outlook for the automatic analysis of dream emotions, which is recognized as a primary dimension of dream construction. With respect to the progression of emotions along the dream reports, our model appears successful at employing the estimates to provide a time course graphical representation.

Table 7. compares the best experimental results, based on each group of attributes individually.

Attributes	Number of attributes after attribute selection	Agreement with Human judge(accuracy)	Mean Square Error (MSE)
Text only ³⁹ (without subjective emotion tags)	39	55%	0.3819
Emotion Prog. Analysis ⁴⁰	21	49%	0.4073
Subjective emotions only ⁴¹	8	48%	0.4269
Union of all above attributes after feature selection	31	64%	0.3617

Table 7. Results of our best classifiers applied on each of the attribute subsets individually.

We also analyzed the estimation accuracy of the eight subjective emotions tagged to the dream questionnaires. Based only on the text representations methods developed in this

³⁸ Literature shows 57 to 80% agreement in human judgment in this area and range.

³⁹ The applied classifier was “bayesNet” from “Bayes” classifier category in WEKA

⁴⁰ The applied classifier was “Multinomial Logistic Regression (Logistic)” from “Functions” classifier category in WEKA

⁴¹ The applied classifier was “Logistic Model Trees (LMT)” from “Trees” classifier category in WEKA

paper and using a Logistic Model Trees (LMT)⁴² classifier from “Trees” category in WEKA, we could predict the four levels of Anxiety⁴² (0-3) with 71.4% accuracy (baseline: 28.8%) and a Mean Square Error (MSE) of 30%. Also, 68% accuracy (i.e. agreement with human judges in the four scales) with the same rate of MSE was obtained for the ‘Fear’ emotion, and for the other emotions the obtained accuracy was not less than 60%.

We believe the results can be improved. Larger databases will facilitate analysis and data mining, and emotion specific parameters could add resolution and improve accuracy. To the extent that dream narrative corresponds to the time-course of dream experience, the graphical representations provide a new tool for exploring models of dream formation.

Further development of this technology could facilitate the analysis and mining of more dreams of individuals of different age, sex and social status, thus improving our overall understanding of dreams.

4.1.6. SUBJECTIVE EMOTION ESTIMATION BASED ON THE OTHER EMOTIONS

Recall that the dreamers were asked to rank their dreams (0-3 range) for the eight given emotions in their questionnaire. It was interesting to see to what degree the emotions were dependent, without taking into account the text content of the dream. Table 4 shows the results for subjective emotion estimations based on the other emotions by percentage (one versus the other seven). For example, the value 97 in the “Happiness” column, shows the level of the happiness in a dream can be estimated (scale 0-3) according to the other 7 emotions (e.g. joy, apprehension, anxiety, etc.), with 97% accuracy. The

⁴² Tagged by dreamers themselves

multinomial logistic regression model with a ridge estimator classifier outperformed other methods in accuracy and mean squared error on the scale 0-3.

	Joy	Happiness	Apprehension	Anger	Sadness	Confusion	Fear	Anxiety
Accuracy	97%	97%	99%	99%	95%	38%	91%	97%
MSE	0.11	0.11	0.08	0.06	0.16	0.44	0.20	0.11

Table 8. Emotion estimation based on the other emotion rates (0-3)⁴³.

The table shows that most of the subjective emotions (except for “Confusion”) are highly dependent to the others and can be accurately estimated by a regression model from the other emotions.

4.1.7. DISCUSSION

In this section we discussed applying the SOSCO text representation method for automatic analysis of sentiments in dreams. 477 dreams were sampled from a dream bank created for a normative study. Expert psychologists assigned scores to dream descriptions based on the expressed sentiments. After applying our SOSCO method on the contextualized dreams, and running several iterations of sentiment analysis, we compared the results. We report the following conclusions:

- Human judgment is based on the text, and his/her perception, background and knowledge, which can differ among people (57 to 80% agreement according to

⁴³ Although joy and happiness are normally considered as same affects in NLP; since the design of the questionnaire and the selected emotions was something which has been already done by the psychology group for some different purposes, hence we decided to keep both.

[Nad06]). The accuracy of our automatic classifier (64%) is close to the human judging average agreement (69%) in the range.

- Machine learning allows automation of human judgment with an accuracy superior to the majority class choice.

In the next section, we discuss the purpose of the second research project, which is to reduce the workload of human experts who are building systematic reviews of medical articles for use in evidence-based medical research. We demonstrate how the first step in systematic review development is equivalent to a text classification task. We use a committee of classifiers to rank biomedical abstracts, based on the predicted relevance to the topic under review.

4.2. Classifying Biomedical Abstracts Using Collective Ranking Technique

Systematic review is a structured process for reviewing literature on a specific topic, with the goal of identifying a targeted subset of knowledge or data. Usually, the reviewed data includes titles and abstracts of biomedical research articles that could be relevant to the topic. The source data was extracted from biomedical literature databases (e.g. MEDLINE⁴⁴) by running queries with keywords selected by domain experts⁴⁵. The queries are intentionally very broad, to ensure that no relevant abstracts are missed, and the output included approximately 10^4 articles. A systematic review can be seen as a text classification problem with two classes: a positive class with articles relevant to the topic of the review, and a negative class for articles that are not relevant.

The selected approach is based on using committees of classification algorithms to rank instances, based on their relevance to the topic of the review.

⁴⁴ Available at: <http://medline.cos.com>

⁴⁵ All the abstracts already include certain keywords in the domain; hence, these keywords can no longer be used for the domain discrimination.

Experiments were performed on a systematic review data set provided by the TrialStat Corporation⁴⁶. The source data included 23,334 medical articles that were pre-selected for the review. While 19,637 articles had a title and an abstract, 3,697 had only the title. The data set had an imbalance rate (i.e. the ratio of the positive class to the entire dataset) of 8.94%.

A stratified, repeated random sampling scheme was applied to validate the experimental results. The data was randomly split into a training set and a test set five times, and the test set representation files were built based only on the training set feature space; features which only occurred in the test set were ignored. For each split, the training set included 7,000 articles (~30%), and the test set included 16,334 articles (~70%) The results from each split were then averaged.

We applied two data representation schemes to build the document-term matrices: BOW and SOSCO representation. CHI2 feature selection was used to exclude terms with low discriminative power.

The ranking approach enabled selecting abstracts that were classified as relevant or non-relevant, with a high level of prediction confidence (i.e. not less than the average prediction performance of human experts).

We aimed to achieve a high level of recall and precision in the Positive class. Applying a range of contrast parameters over all the data would not help us achieve the acceptable level for both recall and precision. For this reason, in our classification process we decided to focus on two tails of certainty: the certainty of being Positive or of being

⁴⁶ Available at: <http://www.trialstat.com/>

Negative. Hence, we created a dataset consisting of only the two tails of certainty, and put the rest aside⁴⁷. We observed that the highest-contrast BOW representation performed well on the segment with high certainty for the Positive class (700 abstracts), while the SOSCO representation, with a contrast parameter value of zero, had better performance on the segment with high certainty for the Negative class (8,000 abstracts).

Therefore, the prediction zone consisted of 8,700 articles (700 top-zone articles and 8,000 bottom-zone articles) which represented 37.3% of the entire corpus (53.3% of the test set). The gray zone included 7,634 articles (32.7% of the corpus) left for human experts to classify; this can save considerable time, since systematic reviews are usually done manually. A committee of five classifiers was applied to the BOW and the SOSCO representation, individually, and the results were then combined through a voting scheme.

The results after voting are presented in Table 9. The table is a confusion matrix, where only the prediction zone articles are considered. Positive articles included in the top zone are true positives (TP), while positive articles included in the bottom zone are false negatives (FN). Negative articles in the top zone are false positives (FP), and negative articles in the bottom zone are true negatives (TN).

Zone	Number of Abstracts	Correctly Classified	Incorrectly Classified
Top	700	590 (TP)	110 (FP)
Bottom	8000	7946 (TN)	54 (FN)

Table 9. Confusion matrix on the prediction zones applying ensemble BOW and SOSCO by voting in a committee of classifiers.

⁴⁷ The legitimacy of working only on the two tails of certainty was based on reducing about 50% of the human expert workload.

Table 10 presents the recall and precision results calculated for the Positive class (the class of interest), based on the prediction zone confusion matrix from Table 9. Table 10 also includes the average recall and precision results for human expert predictions (considered individually). It shows that our method achieved a significant workload reduction (37.3% as mentioned), while maintaining the required performance level.

Performance Measure	Machine Learning performance on the prediction Zone	Average Human Reviewer's performance
Recall on the Positive Class	91.6%	90-95%
Precision on the Positive Class	84.3%	80-85%

Table 10. Performance Evaluation

We verified the performance of using ensemble learning methods over data representation techniques. We chose a committee of five classifiers chosen from the classification library of the Weka package, including: Complement Naïve Bayes; Discriminative Multinomial Naïve Bayes; Alternating Decision Tree; AdaBoost over Logistic Regression and AdaBoost over j48 decision tree algorithm. We used Voting Perception as a meta-classification algorithm based on the above committee of classifiers and ran it first on the SOSCO representation, then ran it on the BOW data representation and, finally, we ran the schema on both data representations together. The results are shown in Tables 11 and 12. The number of misclassifications of both False Positives and False Negatives is significantly less for the ensemble of data representation techniques, than for either of them used alone. In the following tables, we can see the performance of each representation at each tail of certainty.

Split Number	SOSCO	BOW	Ensemble (SOSCO and BOW)
1	138	127	110
2	138	186	108
3	118	117	101
4	143	119	113
5	160	130	119
Average	139.4	135.8	110.2

Table 11. False Positives with respect to data representation methods. Numbers are out of the 8,700 articles in the process.

Split Number	SOSCO	BOW	Ensemble (SOSCO and BOW)
1	55	78	53
2	55	119	48
3	55	96	55
4	71	72	50
5	68	101	62
Average	60.8	93.2	53.6

Table 12. False Negatives with respect to data representation methods.

Since the machine learning prediction performance is generally equivalent to the human prediction performance, using the system will lead to significant workload reduction for the human experts involved in the systematic review process.

In the next two chapters we will transform the SOSCO representation to some new, lower-dimensional representations. We show that, among the new features of the new representations, there are some information granules that can potentially help us to elaborate our ontological representation (as ontological nodes), and improve the representation power of the targeted learning domain. These new features are the *conceptual elements*, which have already been introduced as *shared vocabulary* in section 2.3.

5. Automatic Text Ontology Representations

5.1. Overview

The idea here is to represent corpus entries ontologically, based simultaneously on their context and the goal of the learning task. This is unlike the traditional representation methods, in which all the entries are represented based solely on their texts, totally isolated from the goal they are represented for.

In this chapter, we introduce an automatic method to extract a lightweight ontology from a variety of candidate conceptual elements acting as ontology nodes (i.e. groups/classes in the literature), in different levels from fundamental to specific (in Figure 18 we attempt to depict an abstract idea which will be developed in the example of “*your favorite car*” in the next section). To do this, we attempt to create a fusion of the candidate conceptual elements, to achieve the most discriminative ontology over the target concept of our learning task. This goal implies reconfiguration of the set of conceptual elements which represent our corpus, if there is a change in the learning task.

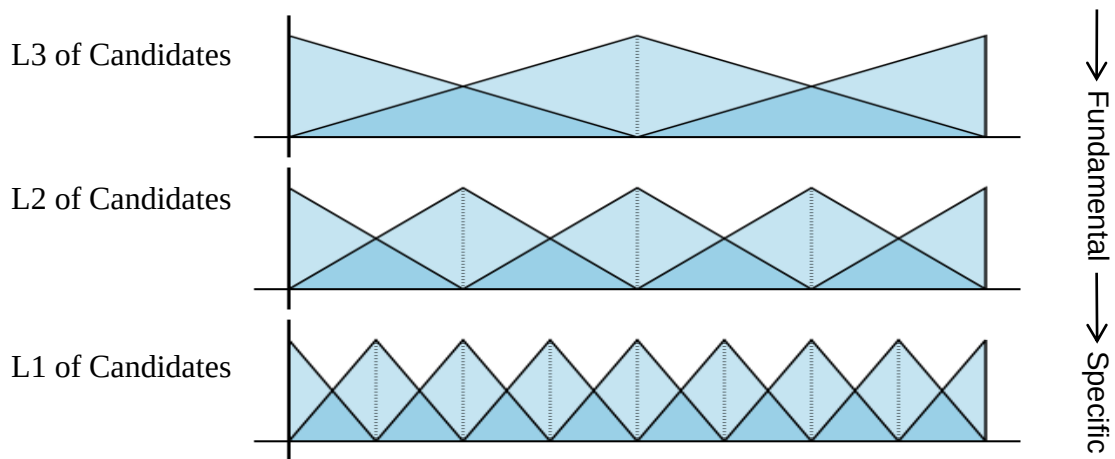


Figure 18. Each triangle illustrates a candidate node (conceptual element) for the ultimate lightweight ontological representation. A candidate ontological node is applied as a new feature for domain representation. The intersections between the ontological nodes (darker areas) depict the existing relationship/dependency between the features (conceptual elements).

The procedure is detailed in the following sections.

5.2. Conceptual Representation versus Low-level Feature Representation

We believe interpreting data in a more general dimensionality, consisting of fewer dimensions (when appropriate), can convey more discriminative knowledge.

David Donoho states that the high dimensionality phenomenon is not always a problem, and might provide new opportunities [Don00]. He says "...although those blessings of dimensionality are less widely noted; however we can take advantage of them to simplify our data analysis " by replacing many "similar" dimensions by a median of them.

We often encounter high dimensional data only because the underlying objects are from a continuum. For example, if there is an underlying curve or image that we are sampling, increasing the number of samples increases data dimensionality (e.g. wavelets). This could be compared to a crossword puzzle; we do not need to know all the answers to complete the puzzle, and knowing some of the answers can provide others. However, sometimes the underlying continuous-space is not well defined (well-known), and we have to explain it with discretized attributes in a relatively high-dimensional space (e.g. text representation).

In other words, if we had all the feature spaces classified into a certain number of independent groups, one representative of each group would be useful to represent the corpus entries. However, in reality natural language words are closely related to each other, and it is impossible to perform such a deterministic classification.

The idea is that we can still reduce the dimensionality of our space with *soft dependency calculation between dimensions*, and represent each point (document) in a new coordinate with lower dimensionality, after performing a proper coordinate transformation process.

For better understanding and clarification of this idea, consider the following example.

Assume we are given a table in which each row contains information items related to over one thousand constituent parts of an individual car, in a certain order. By referring to each row of the dataset, we can get extensive information about a car based on these constituent parts. This information can tell us about *specifications* in detail, including *manufacturing company*, *producing country* and *price* of the parts separately, without indicating the exact name and model of the car. Now, imagine that you are asked to choose *your favorite car(s)*, among the approximately 500 cars in the dataset. That would be difficult, because we normally choose a car based on more fundamental attributes, such as body style (SUV, Sport or Sedan), fuel type (Gas, Diesel, Hybrid, or Electricity), number of cylinders, specification of the engine, specification of the gearbox, specification of the braking system, reputation of the manufacturer company and, finally, price.

Although each of the above coarse granular attributes of the cars are not included in the fine granular raw data, each of them can be predicted from the original dataset as a separate and specific learning task in order to boost the performance of our ultimate search. Therefore, it could be effective to extract some higher granular attributes of the cars, and then use the membership information of those fundamental attributes to predict the final answer; which is concerned with the correct level of information granularity of attributes that should represent the original data.

To apply the above approach in a textual domain, it would be helpful to have the corpus and its domain ontology at the same time to have a sense of its coarse granular attribute as well as its vocabulary features as fine granular attributes. Since in most text classification cases we are not given the domain ontology, thus it could be helpful to build at least a lightweight ontology in prior to run any classifier over the corpus. To achieve the explained goal we will extract a variety of conceptual elements as our candidate coarse granular information items (as nodes of the lightweight ontology) from a stack of fundamental to specific text representations, and combine them according to their discrimination power on the ultimate learning concept.

5.3. Extraction of the Conceptual Elements (Candidate Nodes of Ontology) via Clustering

In order to recognize the most informative and expressive set of concepts, we reviewed the literature (see the Background Section) and decided to apply the most appropriate clustering method for our inherently high-dimensional textual domain.

Ideally, the clustering method would apply the subspace clustering algorithm, (explained in Section 2.6.7.) as our approach to the curse of dimensionality. Hence, in order to find the best subspaces we need a way to measure of strength of relationship for pairs of words. This measure is shown and explained by the definition of the Mahalanobis distance method in section 2.6.1. Thus, in our SOSCO representation, to have a meaningful distance measure we consider the inter-relationship of the corpus words (Closeness Matrix that implements a Mahalanobis-like distance measure). We have introduced a hybrid-clustering (Section 2.6.9.) approach, in which clusters (conceptual elements) emerge from the feature space/subspace based on the inter-relations that have

been defined in Section 3.3. In our method, the candidate nodes of our light-weight ontology are directly extracted from word representative vectors through a fuzzy clustering process.

We applied some candidate clustering/distance combinations on bag of words and SOSCO representations of text data, to observe and get a sense of the quality of the outcome clusters.

The following clustering method has been configured to perform our conceptual clustering task:

- Since we already consider the inter-relationship between features in our base word-level representation empirically, a simplified version of the Minkowski distance function ($p=1$; which is called Manhattan distance function) was selected. This measures the distance between two individual words or possibly the distance between a word and the centroid of a cluster, depending on the Linkage type (e.g. Single-linkage). See section 2.6.2.
- The distance between clusters (e.g. for neighbor joining) is interpreted as the *branch-length* (in the hierarchy). We choose this measure empirically⁴⁸ to get better performance from soft hierarchical clustering over the feature (word) vectors created in section 3.3.
- The soft hierarchical clustering is implemented using a density-based cluster distributor wrapper, which encloses the main hierarchical clustering core. The wrapper

⁴⁸ In Weka stable version (3.6.4.) when we use the hierarchical clustering, we have to set a parameter named "Link type"; since in the very beginning stages of concept extraction via clustering (not on this dataset) we noticed that "neighbor joining" value for this parameter works faster with not considerable jeopardize of the performance, therefore we decided to use it for all the experiments mentioned in this thesis.

not only imposes a certain minimum cluster density, but also selects only the dimensions (words) for which the intra-cluster standard deviation does not exceed a given threshold (for each cluster that emerges by the wrapped clustering method). In other words, to solve the problem of the *curse of dimensionality*, during the process of cluster distribution assignment for document vectors (soft clustering in section 2.6.2.), the wrapper nullifies the effect of dimensions with high standard deviation as noise.

- The characteristic of any short document $d_i \in \mathcal{D}$ is defined based on the relevancy distribution of the entire set of the conceptual element $c \in \mathcal{C}$ to d_i .
- We represent each document $d_i \in \mathcal{D}$ in some conceptual level $\mathcal{C}$, where: $d_i = \{\bar{r}(c, i)\}_{c \in \{1, \dots, |\mathcal{C}|\}}$ and $\bar{r}(c, i)$ is the normalized membership degree of d_i to the cluster c , as a candidate conceptual element in $\mathcal{C}$.

In other words, we define an ambiguous term (word) clustering algorithm with words that belong to more than one cluster only if they are related to different conceptual elements $e \in E$, and the term belongs to at least one of the soft conceptual clusters to some degree. In more precise notation, we define a set of conceptual elements, $E_{\text{ambiguous}}(w)$, for each $w \in V$ in our vocabulary V , as:

$$E_{\text{ambiguous}}(w) = \{e \in E \mid \|w - \mu_e\| \leq d_w + \epsilon\}$$

where d_w represents the distance of w to its nearest cluster centroid μ of the conceptual element e in E (μ_e), and ϵ is a pre-set *maximum-range* parameter. Consequently, word w would be an *ambiguous* word if $|E_{\text{ambiguous}}(w)| > 1$.

We recall that, we employ a parameter for the *intra-cluster-maximum-standard deviation* that also filters some dimensions (words) in our soft clustering procedure. We assign each document—depending on the degree of membership to each cluster—to one or more

clusters simultaneously; we call this soft clustering. We chose the hierarchical clustering approach as the core clustering process, to provide the hierarchical structure of our ontological representation.

In the next section, we present the entire flow-chart (rough) and algorithm (detail) of the TOR-FUSE representation and classification, and continue our description of the specifications of the methodology over the next two chapters.

5.4. Fundamental to Specific Extraction of the Conceptual Elements

Recall that the SOSCO representation was created according to the extraction of the term-term inter-relationships (multi-configurations) of words over the entire corpus (Chapter 3.). We ended up with a non-sparse representation vector for each document over the corpus, which respects the relevancies over the feature space. Though the closeness matrix represents the lexical relevancies over the feature space, its feature vectors (rows/columns) contain a high volume of redundancies, due to synonymy, antonymy and other types of close feature relationships in high dimensional space. The dimensionality is close to the number of individual words in the corpus, which can potentially reach many thousands. This high dimensionality not only dramatically increases the computational complexity of any further process, but can also inherently be considered as a serious factor in decreasing the performance in many classifiers. Hence, to overcome the expressed concern with the SOSCO representation, instead of including the entire feature space in our representation, we can introduce some expressive

representatives for each group of close enough⁴⁹ features (conceptual elements) as a cluster, and then go through the dimensionality/ambiguity reduction.

5.5. TOR-FUSE Complete Algorithm and Flow-chart

1. Initialization:

I_m = SOSCO output which is a representation of our corpus (input#1);

M_C is the Similarity/Closeness Matrix created by SOSCO (input#2);

m = # of instances (Documents); F_t = the entire feature set; t = # of features in the corpus after stop-word removal; $N = t/2$ is the number of clusters; q = # of classes in the classification task; $S = 1$; Series L_f = null;

2. After clustering the M_C into n clusters, the created clustering model⁵⁰ is called “ mdl ”; (the selected clustering algorithm was already introduced in section 5.3., and implemented in Weka⁵¹);

3. For any individual instance x in I_m (the SOSCO representation of the corpus), we apply the clustering model (mdl), to create the cluster distribution vector t_i ($i = 1$ to N); the new vector t_i is a transformed representation of I_m ; where $t_i \in T_s$ ($|T_s| = m$)

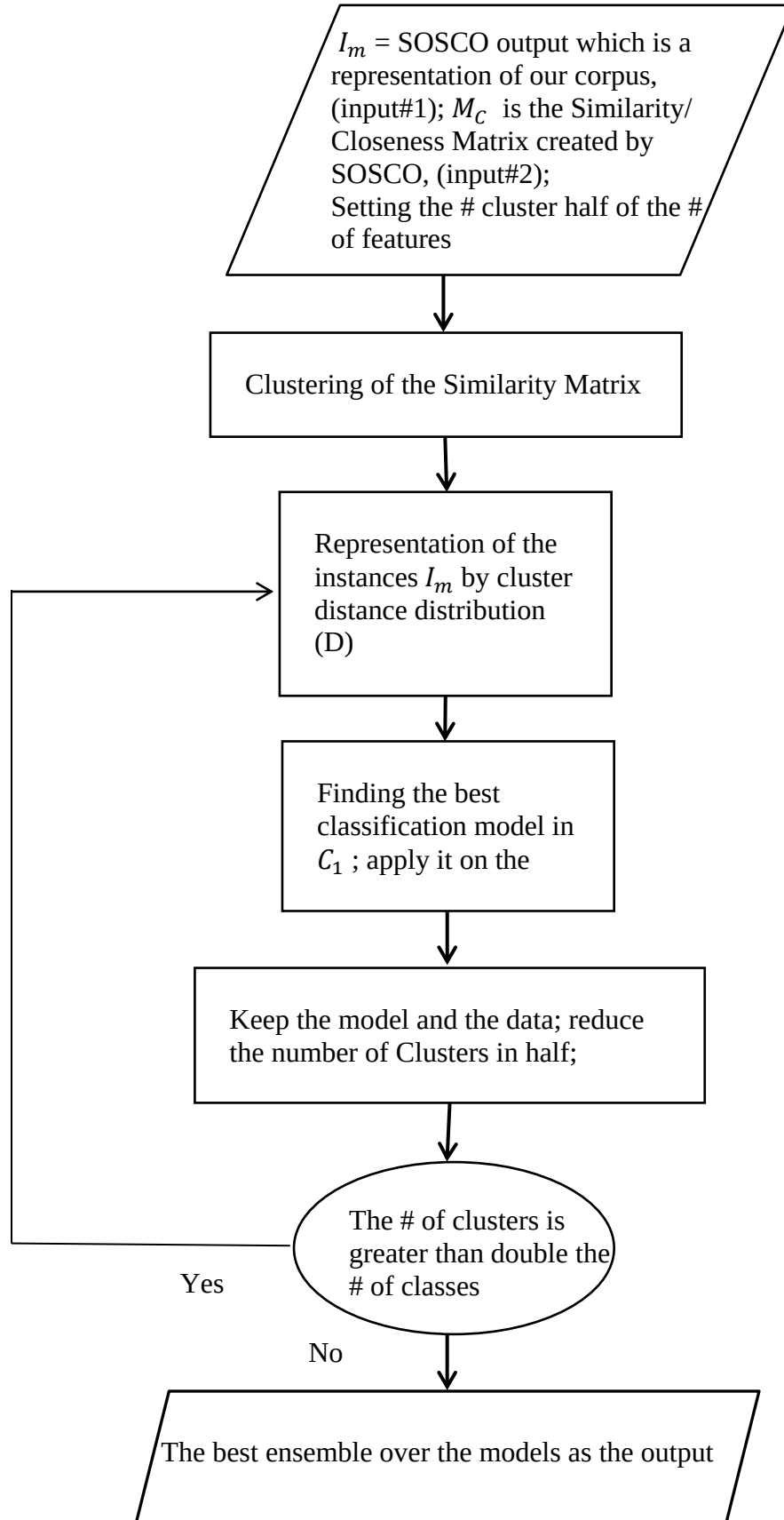
⁴⁹ Identifying enough closeness is related to the target concept to be addressed by the representation (i.e. enough closeness rates would be different when the target concept is agriculture, rather than if the target concept is something more specific, such as irrigation). However, identifying the proper size of an expressive cluster is a major task of the TOR-FUSE algorithm, and will be discussed later.

⁵⁰ The term *Model* is used for a clustering model which is a specific serialized output file to represent any already trained clustering or classification algorithm run in WEKA.

⁵¹ In order to implement the clustering part of the thesis we applied the “MakeDensityBasedClusterer” cluster wrapper class in Weka 3.6.4 standard version; however we modified solely the wrapper complying with the specifications mentioned in the section 5.3 (sub-space feature clustering), the wrapped clustering algorithm is exactly the standard hierarchical clustering class already implemented in Weka.

4. We apply a pre-set of classification methods, $c_1, c_2, \dots, c_{k_1} \in C_1$ (these classifiers must be able to run on high-dimensional data) on the created data set T_s at step 3; then call the most discriminative model⁵² among the k_1 classification models as μ_s ;
5. We use the classification model μ_s to calculate the class distribution probabilities $[p_1, p_2, \dots, p_q]$ of any class label l_j ($j = 1$ to q), for any individual instance $t_i \in T_s$; then we define series $L_s = \langle p_1, p_2, \dots, p_q \rangle$ and L_f as the stack of series L_s (probability distributions), where, $L_f = L_f . L_s$ (dot “.” is the string concatenation operator); we set $S = S + 1$ and $N = N/2$;
6. If n the number of clusters is not less than the number of classification classes (q) then go to Step 2.
7. For each instance x in I_m , let the corresponding $L_f = L_f . Cl_s$, where Cl_s is the real class label assigned in I_m ;
8. We apply a pre-defined set of classification methods $c_1, c_2, \dots, c_{k_2} \in C_2$, (these classifiers are selected to run only on the low-dimensional transformed data) on the set of all L_f (new attribute set) corresponding to all instances in I_m ; the most discriminative learner model (has been defined in the footnote of step 4.) μ_f among the k_2 classifiers would be the output of the algorithm.

⁵² By “The most discriminative model” we mean a model created by a classifier which, 1- can be practically run on the given dataset regarding the number of features/instances (i.e. a logistic regression classifier empirically cannot be run over a dataset with thousands of features and instances); 2- has better accuracy on the dataset running 10-fold-cross-validation; 3- the better accuracy is stable when we run different folding of the 10-fold-cross-validation (to see the stability of a classifier we normally change the “Seed” or random parameter of the 10-fold-cross-validation)



5.6. Text Ontological Representations via Hierarchical Clustering of Features

In this section, we will extract the conceptual elements using the clustering process described in the previous chapter. We described this process with references to its precise presentation in section 6.1. These conceptual elements will be regarded as dimensions/features of a new space, to represent the corpus in a more general way. We may have a certain number of soft clusters in which each feature (word) in the feature space belongs, to some degree, to at least one of the clusters. Since the number of clusters is one of the parameters of most soft-clustering algorithms, at the first level we choose $N=t/2$ as the number of clusters (t is the number of features in SOSCO representation), in order to perform the gradual agglomerative clustering process and prevent aggressive cluster merging, which may cause the loss of meaningful clusters. Therefore, after running the soft clustering algorithm over the word vectors in the main feature space (t), we would have $(t/2)$ clusters for the features (TOR-FUSE algorithm, Step 2). According to the literature, at this stage each cluster can be considered as a small group of similar or very close words in the corpus [Ped05b],[Ped06a],.

After the soft clustering process (TOR-FUSE algorithm, Step 3) we create a cluster membership distribution vector⁵³, with $t/2$ dimensions for each document (instance) in the main feature space. This can be considered a new representation of our documents in a space with $t/2$ dimensionality.

⁵³ A probability vector, which shows the membership probability of a given document to all clusters in the model, and is directly related to the vector distance between the given document to all clusters in the model.

5.7. Stack of Fundamental to Specific Representations

The above process can be performed based on a different number of new coordinates (i.e. $t/4$, $t/8$ and $t/16$ dimensionality, explained in the TOR-FUSE algorithm, Step 5). At this stage, we review a paragraph from section 2.3.: “In order to represent some more general relations over a wide range of documents which are only different in very specific details, we often need to merge specific ontologies into a more general representation. For example, although it is important to know that a text is about harvesting, irrigation, pesticides, siloing, pedology, etc., at the same time it could be important to just know that all of them are about ‘agriculture’.” It is likely that, during transformation of the coordinates (e.g. $t/4$, $t/8$, $t/16$, etc.), there are some close clusters which merge to create new generic coordinates with lower dimensionality.

We perform the transformations iteratively, to compose a variety of new representations (multi-level) of the corpus based on the new coordinates, with an exponentially more fundamental and lower number of dimensions.

Each new dimension of any individual level can be considered a potential conceptual element/node of candidate ontology over the entire corpus. Since, empirically, the number of attributes of text representation should not be less than 200 to 300 at the most fundamental representative level, the number of levels will not theoretically exceed 7 or 8⁵⁴. By reducing the number of dimensions through the stack of representations of the corpus, the candidate conceptual elements in higher levels could potentially explore more

⁵⁴ $200 \times 2^8 = 51200$ (features); since we initially targeted short text documents corpora (for our algorithm), It is hard to find such a corpus with more than 51200 features. (e.g. Reuter 21578 has less than 30000 individual features for the classification task); however adding one or two level to the TOR-FUSE does not lose the generality of the used method.

fundamental concepts (versus specific details) of the corresponding corpus (Figure 19 assists us to convey the idea).

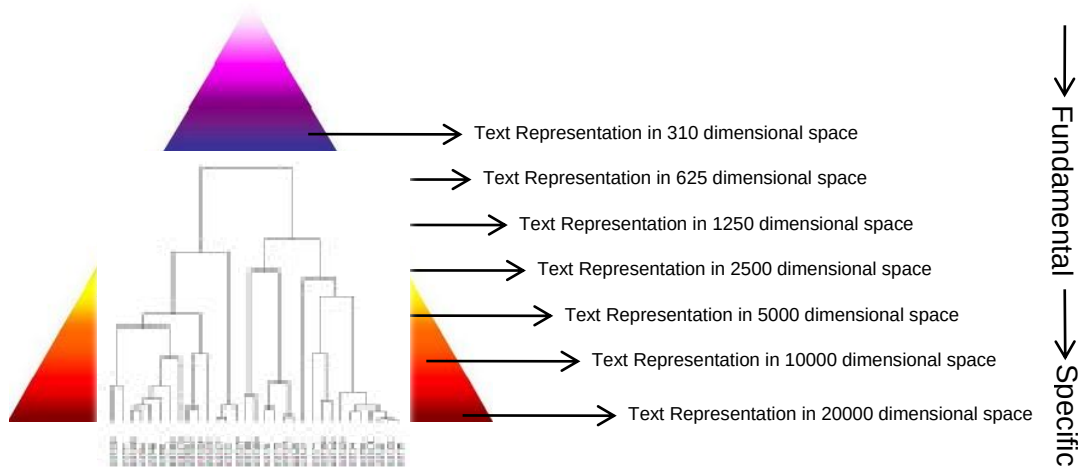


Figure 19. Targeted concept may be modeled /extracted through the nested multi-level TOR-FUSE

5.8. TOR-FUSE – Classification process (Ensemble learning)

We now introduce a tailored ensemble learner, in order to explore the most descriptive composition of ontological nodes (conceptual elements). This is done by running an ensemble learner (Stacking) on the multi-level stack of representation that was created in the previous chapter; hence, a general literature review on ensemble learning is appropriate, and can be found in Chapter 2.

In this chapter, we explain how we built our ensemble learner to explore the light-weight ontology according to the best performing related algorithms in the literature (Section 2.7.). We built a *fusion of classifiers*,⁵⁵ in which the number of individual classifiers is the same as the height of the already generated TOR-FUSE stack of representations (Fundamental to Specific conceptual element sets) for the learning task.

⁵⁵ Applied method is a tailored version of the Stacking method that was introduced in Section 2.7. and its performance empirically was evaluated in a separate application. For more details please refer to Appendix I.

5.8.1. LEARNING OVER THE FUNDAMENTAL TO SPECIFIC ONTOLOGICAL LAYERS

Assuming we have a fundamental to specific stack of potential candidates of ontological representations (over a certain domain) from Chapter 6, we follow two steps for classification or labeling over the instances of the given data:

We recall that we start from the SOSCO representation at level one of the stack of TOR-FUSE. In the first step, for each level of the stack we need to find the most descriptive model over the corresponding conceptual elements, based on the given labels of our class attributes. This model is a proper composition of the nodes of light-weight ontology of that level, and is done as a competitive classification task by a committee of pre-set classifiers that are likely to perform well in a high-dimensional space.⁵⁶ However, if we make some changes to the definition of our classification task (reflected by the class labels which were given initially), but not to the corpus to be represented, the changes could result in extracting a different composition model of the ontological nodes, specifically to explain the new targeted concept for the new classification task.

For the second step we need to create a second level ensemble learner, to go over the assigned probability distribution for the class labels from the first step classifiers as its own attributes, and build the best combined model of the labels as the final labels to be tagged on the instances.

⁵⁶ Our pre-set classifiers for this level are Naïve base multinomial, Complementary Naïve base, SMO (SVM) and Discriminative Multinomial Naïve Base for Text (DMNBText) and K-nearest neighbours classifier (IBK) all applied in Weka.

Thus, the ultimate light-weight ontology is mapped in two ways: First to the conceptual-elements in each level, and second to the levels of ontological representations. This structure is entirely affected by the way labels are assigned to our class attribute.

Since ontological nodes (conceptual elements) of the stack of transformed representations (from Chapter 6.) have been extracted via consisting hierarchical clustering procedures, the ultimate light-weight ontology can be envisioned as a hierarchical structure.

Though choosing the individual classifiers could be done either homogeneously (same classifiers, different parameters) or heterogeneously (different classifiers), corresponding to the characterization of attributes of the stack—which varies between specific details and fundamental trends with high to low dimensionalities—the more rational configuration would likely start from some low computational complexity⁵⁷ at the high dimensional levels, and gradually go through the more complicated algorithms at the lower dimensional levels. At each level, the corresponding individual classifier (stacking classifier) attempts to benefit from each attribute, based on its level of discriminative power for the classification task (first step). In other words, among the attributes of each level of TOR-FUSE, there are some attributes which are more influential for the classification task. Although, in the upper TOR-FUSE levels all the attributes may be considered as the fundamental dimensions of the domain (input data), some have higher discriminative power than others for the learned concept.

⁵⁷ e.g. Multinomial Naïve Base, or some version of SVMs, with relaxing assumptions which have considerably better performance over high dimensionality representations in an acceptable execution time.

After the selection of the stacking classifiers (first level) over the layers of TOR-FUSE, the output of those classifiers (probability distribution on the class values) will participate in the ultimate ensemble classification process (the second level of our stacking method), based on their individual performance over the training set. In other words, there is a proper weighting process over the output probability distribution for class values of stacking classifiers, based on their performance over the training set. Thus, the label assigned to a given instance by a classifier with higher accuracy will get a higher weight, compared to labels which have been assigned by a lower performance classifier (second step).

At this point, the classifiers are combined (Stacking - ensemble learning) according to the output labels associated with their corresponding weights, to predict the ultimate label for each instance in the dataset. This combinatory process will be performed as a classification competition by a variety of next level classifiers including: Logistic Regression (Logistic); SVM (SMO); AdaBoost on J48, decision table; naive Bayes hybrid classifier (DTNB); Logistic Model Tree (LMT) and the J48 all extracted from the Weka classification library.

Through the steps of ensemble learning, we observe that the classification models extracted from the TOR-FUSE levels with higher discriminative power will participate in the final classification proportionally more effectively than the other levels. In each level, the same logic applies for the attributes with more discriminatory power over the targeted learning concept for the classification task.

The above ensemble learning procedure can be visualized as a two dimensional geographical map of a territory, on which higher land is more prominently depicted than

other land or seas. Therefore, we call the fusion method a two dimensional modeling of the targeted learning concept over the layers of TOR-FUSE.

6. Experiments and Results⁵⁸

In order to have further evaluation on SOSCO representation and the TOR-FUSE algorithm, we conducted numerous experiments and evaluation processes on three textual datasets which are publicly available and can be used and compared in the future. The first set of experiments was run on a regular topic/subject classified dataset. Since these types of tasks rely mainly on domain specific keywords, we decided to run two other sets of experiments on data-sets for sentiment detection tasks. The main difference between the two selected datasets is the distribution of the sentiments in the data: in the first dataset we have balanced distribution over the class labels, while in the second the distribution is unbalanced.

6.1. Reuters Transcribed Subset

The first dataset we chose to run our experiments on was the Reuters Transcribed Subset. This is a selection of 20 files from each of the 10 largest classes in the Reuters-21578 collection (see e.g. the UCI machine learning repository⁵⁹), a typical text dataset benchmark.

The source document collection appeared on the Reuters newswire in 1987, and was assembled and categorized by Reuters personnel.

The data includes 10 directories labeled by topic name, each containing 20 files of transcriptions (except for the 'trade' directory, which has 21 files). The topics that will be

⁵⁸ Most of the results included in this chapter has been calculated by "SHARCNET" high performance computers; For more details please refer to: <https://www.sharcnet.ca/my/about>

⁵⁹ <http://archive.ics.uci.edu/ml/index.html>

considered as class labels in our experiments are acq, corn, crude, earn, grain, interest, money, ship, trade and wheat.

- We applied a variety of Bag Of Word (BOW) representations (i.e. binary, frequency and TF-IDF based methods) in order to create the best discriminative representation over the training and testing set on the entire feature set. For evaluation and comparison purposes, a set of stop-words removed form of the TF-IDF based representation was selected. For the BOW representation only we applied Snowball⁶⁰ stemming algorithm on the feature space which includes 5480 words.
- We transformed the entire BOW feature space using Latent Semantic Analysis (LSA), and evaluated the output data as another representation (BOW+LSA). We use 10-fold stratified cross-validation for evaluation of this dataset—this is more feasible when the number of instances are limited— we used WEKA to perform the LSA transformation; the best output data we could extract from our source data includes 170 transformed dimensions.
- We then ran the SOSCO extraction program to get its second representation, which consisted of 2,793 features (words).
- The last step was to perform the TOR-FUSE process (see Section 5.4.) to acquire our last representation and comparisons.

⁶⁰ For more information please see: <http://snowball.tartarus.org/>

To conduct our empirical performance evaluation of a supervised machine learning algorithm, it was necessary to have two disjoint subsets: training and testing. We used the training subset to train our classification algorithms and generate a hypothesis (or a model in Weka). The test subset was applied to measure the classification performance, such as the percentage of examples correctly classified by the hypothesis. In most cases, increasing the number of training samples improved the performance of the learning task. Partitioned training and testing datasets can provide reliable results only when we have enough samples to split into large enough subsets for the training and testing processes. Normally, when we do not have enough instances to split into large enough training and testing sets, we evaluate our classification process based on stratified 10-fold cross-validations. This means that we split the entire dataset into 10 almost equal size and class distribution folds, then train a classifier 10 times on a different 9 fold integration of the entire 10-folds, and test it on the 10th one.

In this case, since the dataset only consisted of 201 short documents in 10 class groups, we performed our evaluation process using the 10-fold cross-validation method⁶¹.

We conducted an extensive number of experiments on a range of classifiers and parameters for each representation; to check the stability of a classifier performance we normally change a small number of times, the “Seed” parameter of the 10-fold-cross – validation to avoid the accidental “over-fitting”; the results, which appear in the following table, are the best stable of all⁶². Looking top-down through the evaluation

⁶¹ We are planning to run the same experiments using five time 2-fold-cross validation schema so that the size of the test sets would be significantly larger; i.e. %50 of the data.

⁶² In order to resolve any conjecture of over-fitting that may arise from selecting a single best fitting classifier is steps 4 and 8 of the TOR-FUSE algorithm, the final evaluation of the method has been

measures we can see how much improvement we achieved when the representation is changed. Since the 10 class labels (topics) are distributed evenly over the Reuters Transcribed Subset data, the baseline for any classification experiment over the dataset may be considered as 10%. We found the Precision, Recall, F-measure, Accuracy, True Positive (TP) and False Positive (FP) rates (the most common and declarative evaluation measures recently used in most machine learning papers), and calculated their weighted average (Wtd. Avg.) value for our experiments. For example, the weighted average value of "Recall" is calculated by averaging the recall of each class value, weighted by the percentage of that value in the test-set. However, in this dataset the class values are almost evenly distributed in all training and testing subsets. Each of the evaluation measures that appears in the result tables (from this point on) is a macro average of 6-10 class label values multiply to 10 folds of one run of 10-fold-cross-validation. For example the "Recall" measure is the average of 6-10 class labels "Recalls" and on the top of that each class labels recall (e.g. Happiness) indeed is the Macro average of 10 Recall values calculated for the runs of 10-fold-cross-validation. Since in most cases the distributions are balanced so averaging over 60-100 number tends to stay at the mean value of their range. All of the averaging process has been done by the standard version of the WEKA 3.6.4.

The extracted values for each of the evaluation measures are shown in the following table⁶³.

performed on a set of four pre-set (fixed) classifiers; we also ran paired t-test only on the fixed (algorithm and parameters) classifiers; the range of accuracy variance can be seen in the table 16.

⁶³ The percentage of correctly classified instances (accuracy) for the TOR-FUSE levels varied in the range of (%20, %38).

Evaluation measure →	TP Rate	FP Rate	Precision	Recall	F-Measure	Accuracy
Representation/ Classifier used ↓	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	%
BOW+LSA/ SimpleCart ⁶⁴	0.418	0.065	0.412	0.418	0.409	41.791
BOW/MNNB ⁶⁵	0.517	0.053	0.533	0.517	0.516	51.7413
SOSCO/ MNNB	0.507	0.055	0.529	0.507	0.498	50.7463
SOSCO+BOW/Comp NB ⁶⁶ + Feature Selection ⁶⁷	0.562	0.049	0.552	0.562	0.542	56.218
TOR-FUSE/Stacking ⁶⁸	0.557	0.049	0.598	0.557	0.569	55.7214
TOR-FUSE+BOW/ Stacking	0.627	0.041	0.624	0.627	0.623	62.6866

Table 13. Comparison of the chosen evaluation measures of the classifiers for different representation methods. BOW+LSA is the LSA transformation of the explained BOW features; SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE Representation method with the BOW representation as level 0 of its stack (a level lower SOSCO which is in the level one).

6.2. ISEAR Dataset

The ISEAR (Intercultural Study on Emotional Antecedents and Reactions) dataset consists of 7,666 short texts that were collected and used for the first time by Scherer & Wallbott in 1994 [Sch94]. The dataset was gathered from 1,096 participants' memories of their reactions to recent situations in which they had experienced strong emotions, of the

⁶⁴ SimpleCart classifier, used from the decision tree category of Weka toolkit. For more details, refer to Weka documentation at: <http://www.cs.waikato.ac.nz/ml/weka>

⁶⁵ Multinomial Naïve Base; For more details refer to Weka documentation at: <http://www.cs.waikato.ac.nz/ml/weka>

⁶⁶ Complementary Naïve Base; For more details refer to Weka documentation at: <http://www.cs.waikato.ac.nz/ml/weka>

⁶⁷ We applied feature selection and ranked the features inside the 10-fold cross validation process. The best-first 1,500 features were selected for the dataset. The feature selection algorithm was Chi-squared that has been applied via "AttributeSelectedClassifier" meta classifier in Weka. For more details refer to attribute selection methods explained in the WEKA.

⁶⁸ Explained in Section 7.2.

sort specified in each of seven questionnaires. The participants completed the questionnaires to report their emotional experiences and reactions to the seven emotions of anger, disgust, fear, joy, sadness, shame and guilt. The respondents, who were from universities in 37 countries on five continents, had two pages to explain each emotional impression. In another related paper, Scherer claims that ISEAR is the largest cross-cultural dataset on emotions [Sch97]. Many other research studies have been conducted over that dataset, including [Cha11]. In our experiments, however, we use only the 5,477 texts which are in one of the five emotion categories that overlap with the basic emotions identified by Ekman [Ekm78]: anger, disgust, fear, joy and sadness (“surprise” excluded since it was not distributed evenly with the other emotions). These are almost evenly distributed among the data, which means there are from 1,094 to 1,096 instances for each of the five emotions.

We conducted our first set of sentiment classification experiments on the ISEAR dataset, which has enough instances for separate training and testing sets, and continued with the AMAN dataset by applying 10-fold stratified cross-validation, which is more feasible when the number of instances is limited. We initially split the ISEAR dataset randomly into 5 stratified folds of 1095-1096 instances, including 219-220 instances for each emotion. We then merged the first 4 folds, including 4382 instances, as our training subset, and set aside the fifth fold, which included 1096 instances, as the testing subset.

- For that data set initially we made the same BOW representation as explained for the Reuters Transcribed Subset, which includes 7,803 features⁶⁹.
- In the next step, we created the co-occurrence matrix and the SOSCO representation of the training set only, then built the SOSCO representation of the test set based on the training co-occurrence matrix and only the vocabulary of the training set⁷⁰. The size of vocabulary at this stage was 3,299 words, after stop-word removal⁷¹. We could not use the entire feature set, since the co-occurrence matrix did not include the features only appearing in the testing subsets.
- Finally, we created the TOR-FUSE representation, followed by its classification section of the algorithm (section 5.4.)

In all the experiments, we trained a model on the training data (80%) and tested it on the test sets (20%) for final evaluation. To explore further, we conducted several experiments for each representation, using a range of classifiers and parameters. Looking top-down through the evaluation measures we can see how much improvement we achieved when the representation is changed.

Since the five class labels (emotions) are distributed evenly over the ISEAR dataset, the baseline for any classification experiment over the dataset can be considered as 20%.

⁶⁹ The difference between the number of features in SOSCO and BOW is due to: I. using the entire dataset feature space for the BOW versus only the training set feature space for the SOSCO, and II using a dynamic stop-word removal of the SOSCO process.

⁷⁰ We ignored any new words that possibly appeared only in the test fold and had not been in the training subset.

⁷¹ For more detail about the SOSCO representation see Chapter 3.

The following table shows the evaluation measures extracted at each stage of the task.⁷²

Evaluation measure →	TP Rate	FP Rate	Precision	Recall	F-Measure	Accuracy
Representation/ Classifier used ↓	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	%
BOW/SMO ⁷³	0.595	0.102	0.597	0.595	0.596	59.5434
SOSCO/SimpleLogistic	0.606	0.096	0.621	0.606	0.61	60.6393
SOSCO+BOW/Avg. Probability Voting ⁷⁴	0.667	0.083	0.667	0.667	0.667	66.6667
TOR-FUSE/Stacking	0.727	0.068	0.729	0.727	0.728	72.6941
TOR-FUSE+BOW/ Stacking	0.763	0.059	0.764	0.763	0.764	76.347

Table 14. Comparison of the classification evaluation measures for different representation methods. SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE Representation method, with the BOW representation as level 0 of its stack (a level lower than SOSCO which is in the level one).

6.3. The AMAN dataset

The last data we ran the comparison experiments on was the AMAN dataset [Ama07].

This dataset contains emotional sentences collected from blogs. Ekman's basic emotions (happiness, sadness, anger, disgust, surprise, and fear) were used for sentences annotation, in addition to a neutral category. The sentences were labelled by four human annotators, and we considered only those sentences with emotion categories that the four annotators had agreed upon, and excluded neutral sentences. That final dataset consisted

⁷² The percentage of correctly classified instances (accuracy) for the TOR-FUSE levels varied in the range of (%31, %50).

⁷³ We implemented John Platt's sequential minimal optimization algorithm for training a support vector classifier. For more details refer to the Weka documents.

⁷⁴ A "voting" ensemble learner based on the probability average of class labels in the committee. For more details refer to the Weka documents for "Voting" meta classifier.

of 1,290 sentences. The 6 class labels (emotions) are distributed unevenly over the AMAN dataset, in which we had 536 sentences for “happiness”, 173 sentences for “sadness”, 179 sentences for “anger”, 172 sentences for “disgust”, 115 sentences for “surprise”, and 115 sentences for “fear”. Thus, the baseline for any classification experiment over the dataset may be considered as 41.55%, which is the frequency percentage of the major class, “happiness”. We deliberately ran our comparison experiments on this data to get a sense of the performance on unbalanced data as well.

- For this dataset, we initially made the same BOW representation as already explained for the Reuters Transcribed Subset dataset. The BOW representation dataset includes 3,569 words.
- We transformed the entire feature space using Latent Semantic Analysis (LSA), and evaluated the output data as another representation of the data. The best output data we could extract from our source data included 577 transformed dimensions.
- The same for the previous datasets, in the next step, we ran the SOSCO extraction program to get its second representation, which consisted of 1,038 features (words).
- The last step was applying the TOR-FUSE process to get the third representation and comparisons.

Since the dataset only consisted of 1,290 sentences in 6 emotional categories (class labels), we completed all the classification evaluations based on stratified 10-fold cross-validations. As with the other two datasets, we conducted several experiments on a range of classifiers and parameters for each representation, and the results shown in the following table are almost the best and most stable among them. Looking top-down

through the evaluation measures we can see how much improvement we achieved when the representation is changed.

We conducted extensive experiments with the TOR-FUSE algorithm; to check the stability of a classifier performance we normally change a small number of times, the “Seed” parameter of the 10-fold-cross –validation to avoid the accidental “over-fitting”⁷⁵.

The evaluation measures extracted at each indicator are shown in the following table⁷⁶.

Evaluation measure →	TP Rate	FP Rate	Precision	Recall	F-Measure	Accuracy
Representation/ Classifier used ↓	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	Wtd. Avg.	%
BOW+LSA/SMO	0.57	0.179	0.56	0.57	0.557	56.9767
BOW/MNNB	0.532	0.077	0.628	0.532	0.552	53.1783
SOSCO/ MNNB	0.605	0.127	0.601	0.605	0.602	60.5426
SOSCO+BOW/Comp NB + Feature Selection ⁷⁷	0.633	0.109	0.63	0.633	0.631	63.2558
TOR-FUSE/Stacking	0.76	0.122	0.772	0.76	0.757	75.969
TOR-FUSE+BOW/ Stacking	0.782	0.076	0.783	0.782	0.782	78.2171

Table 15. Comparison of the classification evaluation measures for different representation methods. BOW+LSA is the LSA transformation of the explained BOW features; SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE Representation method with the BOW representation as level 0 of its stack (a level lower SOSCO which is in the level one).

⁷⁵ In order to resolve any conjecture of over-fitting, the final evaluation of the method has been performed on a set of four pre-set (fixed) classifiers; we also ran paired t-test only on the fixed (algorithm and parameters) classifiers; the range of accuracy variance can be seen in the table 16.

⁷⁶ The accuracy for the TOR-FUSE levels varied in the range of (%40, %55).

⁷⁷ We applied feature selection and ranked the features inside the 10-fold cross validation process. The best-first 2,000 features were selected for the dataset. The feature selection algorithm was Chi-squared that has been applied via “AttributeSelectedClassifier” meta classifier in Weka. For more details refer to attribute selection methods explained in the WEKA.

As with the other datasets, we ran a range of classifiers on the levels of TOR-FUSE, and chose one that performed well to calculate the probability distribution for the dataset (Section 5.4. Steps 4 and 5). The output data for each level, which included the class probability distribution (assigned by a classifier⁷⁸ as a new attribute) in addition to the corresponding class labels, were used to run a variety of classifiers.

The percentage of correctly classified instances (accuracy) varied in a range of (40% to 55%). However, only by integrating the above attributes into a united file (final TOR-FUSE file) and running the same classifiers (Section 5.4. Step 8), gave an outcome classification performance of approximately %78. This increase can be interpreted as diversity in information granularity, resulting in a synergetic composition that boosts the performance of the classification task. In order to resolve any conjecture of over-fitting during the process of selecting the best stable classification model, the final evaluation of the method has been performed on a set of four pre-set (fixed) classifiers; we also ran paired t-test only on the fixed (algorithm and parameters) classifiers. Although, paired T-Test is normally applied for comparison between two algorithms or datasets and cannot be considered as a perfect evaluation tool for our case, but in order to have a more extensive evaluation of our representation algorithm, we conducted a memory and time consuming t-test, over all the 24 combinations of the above mentioned datasets and four well-known classification algorithms (i.e., Complementary Naïve Base, Multinomial Naïve base, SMO (SVM) and J48 decision tree). There are other algorithms with potentially higher performance (e.g., Logistic Model Trees or LMT) on low-dimensional

⁷⁸ We did not include any error rate information items (e.g. Output error flag, Mean absolute error, Root mean squared error, Relative absolute error and Root relative squared error, etc.)

datasets like the final TOR-FUSE dataset, but since they cannot be applied on the other high-dimensional data we excluded them in our test. The certainty level %95 has been chosen for the 20 t-tests abstracted in the Table 6. We performed the test with 10 runs for each combination of the dataset and classification scheme. There are 24 combinations and 20 t-test comparisons in the Table 6; each run performs stratified 10-folds-cross-validation, which means 100 total runs for each combination (2,400 runs of 10-fold cross-validation). The following table shows the extracted results, based on correctly classified percentage; looking top-down through the evaluation measures we can see how much improvement we achieved when the representation is changed.

Representation →	BOW+ LSA (Test base)	BOW	SOSCO	SOSCO+ BOW	TOR- FUSE	TOR- FUSE+ BOW
Classifier used ↓						
Comp.NaiveBase	53.83±3.55	59.39±4.03	58.23±3.83	66.45±3.83	66.35±4.02	70.04±3.86
Area under ROC	0.70	v 0.79	v 0.73	v 0.83	v 0.75	v 0.78
MultinomialNB	41.55±0.38	52.98±3.64	59.61±3.80	63.90±4.30	50.93±2.85	53.99±3.12
Area under ROC	0.74	v 0.83	v 0.75	v 0.82	v 0.84	v 0.87
SMO(SVM)	56.78±4.29	63.67±3.57	55.77±3.50	59.81±3.54	74.88±3.38	80.89±3.11
Area under ROC	0.74	v 0.77	0.74	v 0.77	v 0.81	v 0.87
J48 (Decision tree)	31.88±3.59	43.16±2.05	50.90±4.28	50.22±4.26	74.95±3.61	79.24±3.34
Area under ROC	0.57	v 0.58	v 0.66	v 0.73	v 0.86	v 0.86
100 run average	46.01	54.80	56.13	60.09	66.78	71.04
(Win/ Even /Loos)	(0/0/0)	(4/0/0)	(3/1/0)	(4/0/0)	(4/0/0)	(4/0/0)

Table 16. Shows 20 paired t-tests that compare the correctly classified percentage (accuracy) of the four mentioned classifiers over the different representations. BOW+LSA which is the LSA transformation of the explained BOW features is also our tests' base; SOSCO+BOW is the integration of both the SOSCO and BOW features; TOR-FUSE+BOW is the TOR-FUSE representation method with BOW representation as level 0 of its stack; 'v' is the sign of victory or significant difference over the test base (the first column of each row).

7. Discussion

7.1. TOR-FUSE vs. LSI

Latent Semantic Indexing (LSI) claims to overcome the problem of disambiguation by applying its indices as latent conceptual features which are explaining the indexed lexical domain in lower dimensional coordinates. Nevertheless, the dimensions in the so-called latent layers have never been translated to any human-understandable and meaningful concepts and there is only an assumption that there is some underlying or latent structure in word usage that is hidden the original feature space (word meaning space). TOR-FUSE, applies clusters on the main feature space (word); in many researches and applications it has been proved that clusters in different scales normally consist of words reflecting similar meanings, senses and concepts [Ped97b], [Ped98b], [Ban03], [Pat03], [Pur04b], [Kul05b],[Ped07] which are human understandable; However, standard LSI normally transforms the original high-dimensional feature space to a lower dimensional feature space which is not sensible for human being.

Another assumption in LSI is considering a normal distribution of terms (least squares), whereas term frequencies are a count and do not exhibit the normal distribution in documents.

LSI applies SVD over the main high dimensional feature space (word space) and after truncating dimensions with low eigen values assigned (after transition in the new space), it tries to convert all the instances to the new lower dimensional system. Although dimension truncation reduces the complexity of the underling representation, the main point during this procedure is that we do not have any sense of the truncated dimensions; hence, applying any modification over truncation process (e.g. to replace one of the

truncated dimensions with any of the retained dimensions for some reason) needs to run highly time consuming reiterations in order to optimize the dimension subset to be truncated.

The other concern when we choose LSI for reducing the high dimensionality problem is its high computational complexity ($O(n^2)$) over the main feature space.

Sometimes, there is an argument that when we apply LSI over the word feature space, we reduce the storage space for the new lower-dimensional representation (maximum 300 dimensions) comparing with the original (several thousand) word representation, which is generally highly sparse. Hull showed that in reality the opposite is true and we will not get the advantage of storage consumption reduction [Hul94]. However the computational complexity for creating the TOR-FUSE representation in worst case is equal to the maximum cost of clustering or classification algorithm which has been applied in the internal layers of the algorithm since the $O(n)^{79}$ complexity, for the SOSCO part already has been shown in section 3.5.2.

7.2. partial Text Progression Representation (pTPR) - a motivation for future work

Recall the Text Progression Representation (TPR) which has been discussed in section 3.8.; assuming that the entire initial SOSCO feature set is already transformed to the conceptual elements (via chapter 5.) means that the high dimensional feature space of the SOSCO is replaced with the higher levels (lower dimensionality) of TOR-FUSE⁸⁰ conceptual elements.

⁷⁹ where (n) is the number of documents in the corpus.

⁸⁰ About 200 conceptual elements or even less in the last upper level

In the same way the described TPR can be calculated and drawn (in linear time order) based on each ontological node (conceptual element) of TOR-FUSE instead of the low-level feature space of the SOSCO; however the relation between each word in the text stream and a selected conceptual element (instead of the next word in the text) can be calculated via the soft-clustering membership distribution process (see section 5.3.).

The novel TPR representation which is based on selected ontological nodes (instead of words in feature space) may be considered as a partial progression representation (pTPR) based on the selected conceptual element over a $d \in D$ (see Figure 19 to have a sense of the idea).

Considering the creation steps of the partial Text Progression Representation (pTPR), it specifically may be regarded as conceptual text progression representation in which changes of each conceptual element can be observed similar to PLSA which has been discussed in section 2.2. For better understanding of the idea, the pTPR can be envisioned as emerging concepts in our mind while proceeding through (reading) a text document (i.e., the order of the concepts is regarded; see Figure 20.).

We believe this analysis can be used specifically for some NLP tasks such as a text similarity measurement for variety of applications like: relationship extraction, text segmentation and coherent text summarization.

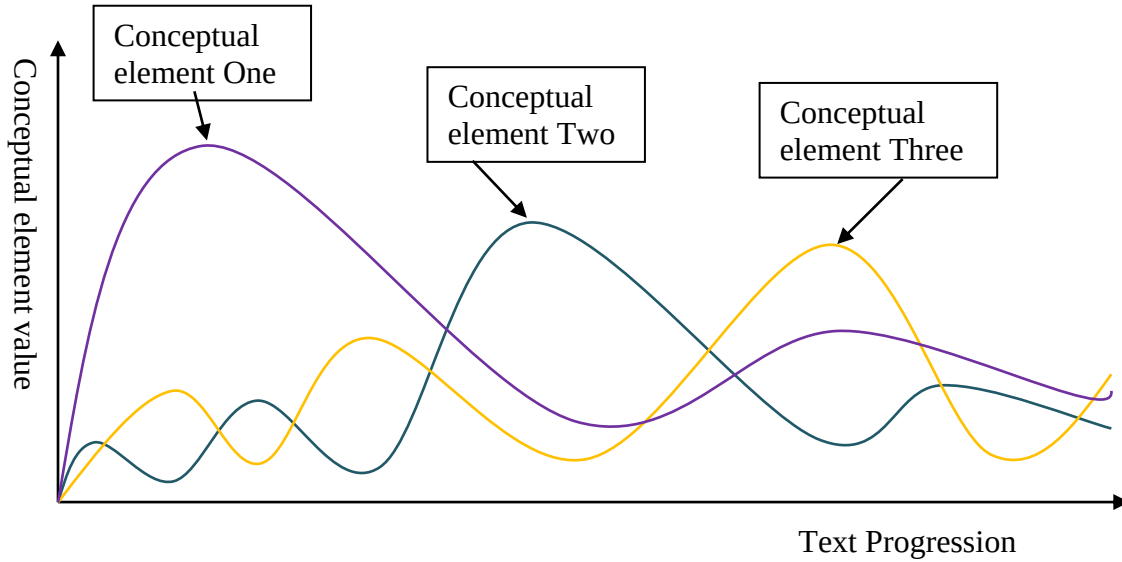


Figure 20. The idea of Partial Text Progression Representation (pTPR) for three sample conceptual elements. Each conceptual element specifically may be considered as a concept which explains the current text over the entire domain. These conceptual elements, may explain the text in different levels of the stack of fundamental to specific layers of TOR-FUSE.

7.3. A Few Notes on TOR-FUSE

- We recall that the subset of TOR-FUSE conceptual elements which is highly-correlated with the learning concept could also be identified by applying some simple feature selection algorithms such as information gain over the entire set of TOR-FUSE ontological nodes (multi-levels). These correlated conceptual elements may be applied as a new set of attributes for a better representation of the corresponding text.
- Since the first set of experiments run on the SOSCO up to the last experiments run on the TOR-FUSE, we boosted our representation performance with integrating them to the BOW representation. We explain that as the “key-words role” in representation. TOR-FUSE representation is based on SOSCO in which

documents are represented by the second order co-occurrence of their words. Thus, in SOSCO (vector) representation of a document not only we have non-zero values for the features (words) explicitly occurred in that document, but also we have non-zero values for the features that co-occurred with the document words in the entire corpus. This means that we suppress the discriminative role of specific key-words in favor of exploring more general concepts from the texts. This phenomenon was our main motivation to define the “Contrast parameter” and is more tangible when we do subject/topic classification rather than sentiment classification.

7.4. TOR-FUSE Limitations

- The computational complexity for creating the TOR-FUSE representation in worst case is equal to the most complex clustering or classification algorithm which has been applied in the internal layers of the algorithm⁸¹. Since TOR-FUSE runs a tailored subspace density-based clustering method which is sensitive ($O(n^2)$) to the number of instances (n) to be clustered, we do not suggest using that in general domain corpora with high-dimensional feature space.
- TOR-FUSE does not demonstrate a tangible ontological taxonomy in its hierarchical structure.
- TOR-FUSE has a small number of parameters to be set (e.g. density parameter and maximum intra-cluster standard deviation). However those parameters has to

⁸¹ The $O(n)$ complexity for the SOSCO part (the first part of TOR-FUSE algorithm) already has been showed in section 3.5.2.

be added to the parameters of the underlying algorithms applied by the algorithm (e.g. clustering and classification algorithms)

- Although we pre-set the number of clusters in each level of the stack of TOR-FUSE half of the cluster number in its previous level, yet we have no optimized decreasing function to make sure we do not miss any meaningful cluster (candidate conceptual element) during the agglomeration process.

7.5. Contributions

The scientific contributions of the thesis are:

- A “second order” text representation method applicable to different conceptual analysis tasks of machine learning domain (SOSCO, Chapter 3).
- The concept of the “Contrast parameter” for targeting different conceptual properties in a context (Section 3.6).
- Text Ontological Representations via FUndamental to Specific conceptual Elements (TOR-FUSE) (Chapters 5 – 7).
- A new representational idea of: Hierarchical vs. Linear Domain Targeting and Concept Analysis (Mapping) for Learning Methods (Section 7.2.).
- Text Progression Representation (TPR) and partial Text Progression Representation (pTPR) tools for text progressive similarity measurement applicable for a variety of open NLP tasks (e.g. relationship extraction, text segmentation, and coherency measure for text summarization (Sections 3.8. and 9.2.).

- Five implemented and published applications for different stages of the proposed representation in the areas of medical abstract selection [Kou09]., offensive language detection⁸² [Raz10], “contrast” in text representation [Raz09], and dream classification [Raz08a], [Raz08b], [Mat10].

7.6. Conclusion and Future Works

In this thesis, we introduced a “second order” text representation method (SOSCO) applicable to different conceptual analysis tasks of machine learning and evaluated it on five experiments such as: Dream data, Systematic review of medical abstracts and three publicly available datasets that already widely used in NLP research including: Reuters Transcribed Subset, ISEAR Dataset and AMAN Dataset. The text Progression Representation (TPR) was introduced and its specific version was applied as dream progression representation, which has been used to extract some progressive similarity attributes in order to improve the corresponding text classification task. We defined the concept of the “Contrast parameter” so we can generate different second order representations to be used for different types of text classification tasks (e.g. topic classification, emotional classification etc). Then we explained a lightweight Text Ontological Representations via FUndamental to Specific conceptual Elements (TOR-FUSE) which implements a novel representational idea of *Hierarchical vs. Linear* concept targeting for text classification tasks. We described three implemented and published applications for different stages of representation boosting methods in the areas

⁸² You can reach the online trial version at: <http://rogets.site.uottawa.ca/Razavi/index.html>

of medical abstract selection [Kou09]; offensive language detection⁸³ [Raz10] and dream classification [Mat10]. The experimental results supported the hypothesis of the “Synergetic power of Generic to Specific hierarchical Text Representations” on the three well-known datasets.

In the future work, in addition to the already mentioned (pTPR), we are planning the following extensions:

- After the soft clustering process (step2 of the TOR-FUSE algorithm), we create a cluster membership distribution vector with $t/2$ dimensions for each word vector (instead of document) extracted from closeness matrix M_C in the main feature space. These distribution vectors can be considered as a new representation of our feature space with $t/2$ dimensionality. Therefore based on the new feature space, the stage of sentence and document representation of SOSCO can be run for the new coordinates ($t/2$ dimensions) and we will have a new representation of our entire corpus with $t/2$ dimensions. This procedure can be performed iteratively with $t/4, t/8, \dots$ dimensionality.
- Add one step of context detection in order to optimize our window size dynamically and build the SOSCO representation vectors based on its component contexts. (Currently our window is based on sentences; see section 3.3.)
- When using SOSCO representation for a text classification task, unlabeled data could be exploited to improve the quality of the closeness matrix M_C which can also be considered as foundation for the TOR-FUSE.

⁸³ You can reach the online trial version at: <http://rogets.site.uottawa.ca/Razavi/index.html>

- Compare the performance in informal/unstructured and formal/structured corpora.
- Implement some more specific representation/learning package to apply for certain sentiments or the level of expressed stress.
- Refine the Dream Progression attributes.
- Explore some specific progression attributes for each sentiment.

Appendix

Multi-level Text Representation and Classification

The initial hypothesis of performance boosting via multi-level text representation and classification was empirically evaluated by the following research project. The hypothesis is considered one of the stakeholders of this dissertation, and plays an infrastructure role for the TOR-FUSE as a synergetic text representation and classification solution. Therefore, in this appendix of the thesis we discuss our initial strong motivation to proceed with an empirical multi-level domain representation and classification case study. The application shows that there are attributes in different granularity levels which can be effective in a learning task.

Offensive Language Detection⁸⁴

Text messaging via the Internet or cellular phones has become a major medium of personal and commercial communication. At the same time, 'flames' (offensive/abusive phrases (e.g. rants, taunts, squalid phrases) which attack or offend users for a variety of reasons) are also transmitted. Automatic, discriminative software with a sensitivity parameter for flames or abusive language detection would be a very useful tool. Although a human could recognize these detrimental types of texts among the useful ones, it is not a simple task for computer programs. In this section, we describe an *automatic flame detection method*, which extracts features at different conceptual levels, and applies multi-level classification for flame detection. While the system utilizes a variety of

⁸⁴ An online demonstration copy of this research can be seen at:
<http://rogets.site.uottawa.ca/cgi-bin/gallery.cgi>

statistical models and rule-based patterns, there is an auxiliary weighted pattern repository which improves accuracy by matching the text to its graded entries.

PRELIMINARY DEFINITION AND DISCUSSION

Often, people must deal with texts (emails or other types of messages) which contain attacks and/or abusive phrases. Thus, automatic intelligent software that can detect flames or other abusive language would be useful, and could save its users time and energy.

Offensive phrases might mock or insult someone, or a group of people (e.g. an aggressive tirade against some culture, subgroup of society, race or ideology). Types of offensive language in this category include:

Taunts: These phrases try to condemn or ridicule the reader in general.

References to handicaps: These phrases attack the reader using his\her shortcomings (e.g. “IQ challenged”).

Squalid language: Phrases which target sexual fetishes or the morality of the reader.

Slurs: Phrases which attack a culture or ethnicity in some way.

Homophobia: Phrases which attack homosexuality.

Racism: Phrases which intimidate individuals' race or ethnicity [Kau00].

Extremism: Phrases which target religion or ideologies.

There are other kinds of flames, in which the flamer abuses or embarrasses the reader (not an attack) by using unusual words/phrases, including:

Crude language: Expressions that embarrass people, mostly because they refer to sexual matters or excrement.

Disguise: Expressions in which the meaning or pronunciation is the same as another more offensive term.

Four-letter words: Five or six specific words with only four letters.

Provocative language: Expressions that may cause anger or violence.

Taboos: Expressions which are forbidden in a certain society/community. Many expressions are forbidden because of what they refer to, not necessarily because there are particular taboo words in the expression.

Unrefined language: Expressions that lack politeness/manners, and the speaker is harsh and rude [Ric91].

Based on the above definitions, when we use the term ‘flame detection’ we are implicitly talking about every context that falls into one or more of these defined cases.

Sometimes, Internet users searching or browsing in some specific sites are frustrated because as they encounter offensive, insulting or abusive messages. This occasionally occurs even in frequently-used websites, like Wikipedia.

Therefore, an automatic system for discriminating between regular texts and flames would save time and energy while browsing the web, and with everyday emails or text messages. Currently, when we review the literature on attempts to discriminate between acceptable contexts and flames, we find considerable disagreement between human expert annotators, even when they have the same definition of flames [Spe97], [Mar02], [Wie04]. Thus, it is evident that we cannot create a definitive and practical product for flame detection for all purposes. Hence, in this paper we will define a tolerance margin for abusive language, based on certain conditions or applications (i.e. different sites and usages), to allow users to have acceptable interaction with computers.

The literature on offensive language detection, and specifically on natural language analysis, describes flames as exhibiting extreme subjectivity [Wie04], depending on the context. This subjectivity is either speculative or evaluative [Mar02]. Speculative expressions include any doubtful phrases, while evaluative expressions deal with emotions (e.g. hate, anger), judgments or opinions [Wie01]. Any sign of extreme subjectivity could be considered a potentially effective feature for evaluation, and flame detection.

However, though computer software does not have the ability to capture the exact concept of a flame context, it does have some useful features, including:

- The frequency of phrases which fall into one of the graded (weighted) flaming patterns (for each grade/weight separately);
- The frequency of graded/weighted words or phrases with abusive/extremist theme, in each grade;
- The highest grade (maximum weight) which occurs in a context;
- The normalized average of the graded/weighted words or phrases.

These features encouraged us to design a fuzzy gauge for flame detection, and implement it in software that could be modified to acceptable tolerance margins, based on training data, manual adjustment, or even instant labeled contexts.

RELATED WORK

Although there are few papers on computerized flame detection methods (which we review in this section), many researchers in Artificial Intelligence and Natural Language Processing have recently worked on different kinds of opinion extraction or sentiment analysis, namely Pang et al. [Pan02], Turney and Littman [Tur03], Gordon et al. [Gor03],

Yu and Hatzivassiloglou [Yu03], Riloff and Wiebe [Ril03a], Yi et al. [Yi03], Dave et al. [Dav03], Riloff et al. [Ril03b], Razavi and Matwin [Raz08a], [Raz08b]. In many cases, detecting the intensity level of moods or attitudes (Negative/Positive) could be an effective attribute of opinion exploration applications for offensive language detection. Furthermore, subjective language recognition could also be useful in flame detection [Spe97], [Wie01]; indeed, flame detection could be considered one of its offspring. In this area, we mention the work of Wiebe and her group: after tagging the contexts (as subjective or non-subjective) using three expert judges, they applied machine learning algorithms for classifying texts based on some of their constituent words and expressions [Bru99], [Wie99]. This study led to similar, but more sophisticated work on evaluative and speculative language extraction [Wie01]. Systematic subjectivity detection could be helpful in flame recognition or email classification [Wie04], [Wie05].

Swearing as a class of offensive language has been studied by Thelwall [The08], who focused mainly on the distribution by age according to gender.

In addition to parts of speech, a corpus can be annotated with demographic features such as age, gender and social class, and textual features such as register, publication medium and domain. However, some abusive languages are related to religion (e.g. “Jesus”, “heaven”, “hell” and “damn”), sex (e.g. “fuck”), racism (e.g. “nigger”), defecation (e.g. “shit”), homophobia (e.g. “queer”) and others. [McE08] and [McE04] examined the pattern of uses of “fuck” and its morphological variants, because it is a typical swear-word that occurs frequently in the British National Corpus (BNC). Also, McEnery et al. in the recent referred article try to expand the examination of “fuck” [McE00a],

[McE00b] by studying the distribution pattern of “fuck” within and across spoken and written registers.

A specific flame detection system we call *Smokey* [Spe97] is probably still being used by Microsoft in commercial applications. Smokey not only considers insulting or abusive words, but also tries to recognize structures of patterns in the flames. It is equipped with a parser for syntactic analysis, which is a preliminary step of a semantic rule-based analysis process. Smokey applies a C4.5 decision tree classifier, which labels each context as a flame or not. At the time of this publication, the system used 720 messages as a training set and 460 messages as a testing set, and achieved a 64% true-positive rate for the *flame* labeled messages, and a 98% true-positive rate for the *okay* labeled messages.

Another method for flame and insult detection is Dependency Structure analysis, which attempts to detect any extreme subjectivity in texts [Mah08].

Unfortunately, *no flame detection software is freely available for trial or research purposes*. Therefore, we cannot directly compare our results to the results of other systems on our dataset.

FLAME ANNOTATED DATA

In this study, we consider a message is a flame if the main intention is *attack* (as described above), or if it contains *abusive* or *hostile* words, phrases or language, according to the desired tolerance margin.

We used two different sources of messages. The first set of data was provided by the NSM (Natural Semantic Module) company log files, and contained 372 sentences in which the company's users ask for some kind of information, service or fun activities, in

an interactive manner. An example of the first dataset offensive statement is: “Do you have plans for this smelly meeting that is supposed to take place today?”

The second set of data we used consisted of 1,288 *Usenet newsgroup messages*, which had already been annotated and used for a flame recognition task by Martin *et al.*[Mar02]. This dataset is balanced across the alt, sci, comp and rec categories from the Usenet hierarchy. An example message that is annotated as a “flame” is: “Feudalist has a new name. How many is that now? Feudalist. Quonster. Backto1913. That’s four with BacktoTheStoneAge. I have never met anyone this insecure before. Actually, I think that BacktoTheStoneAge is intended as a parody. If not, he vastly miscalculated, because I have been laughing hysterically at these posts.” Another example, which is also a “flame” is: “Do you find joy pouncing on strangers I have never found her doing this. Eric, have you?” After deleting messages longer than 2,500 characters, and two which were in French, we were left with 1,153 usable messages. The first dataset is composed mostly of small sentences using abusive language, while the second contains rather long sentences with sarcasm and ironic phrases. We decided to combine them in order to see the performance over a generic and typical offensive language detection task, rather than a specific category.

We used a total of 1,525 messages (1038 (68%) *Okay* and 487 (32%) *Flame*), from both datasets, of which 10% was used as a test set, and the rest as a training set for our multi-level classifier.

METHODOLOGY

After data pre-processing,⁸⁵ we ran a *three-level classification* for flame detection. Considering the attributes of each level, we tried most of the applicable machine learning algorithms implemented in *Weka* (the standard machine learning software developed at the University of Waikato) [Wit08]. We considered factors such as efficiency and updatability for online applications, to determine which classifier to use (e.g. for the first level we needed fast algorithms which could work with a large number of attributes in an acceptable time). After determining which algorithms met these requirements, we chose the one that achieved the highest level of performance among the varieties of simple and combined complex methods available in *Weka*. This process for classifier selection was applied to the other levels as well. The classifiers discussed in this paper provided the highest discriminative power, compared to the other classifiers we tried. In the third level of classification, we use our Insulting and Abusing Language Dictionary, which contains some words, phrases and expression patterns for corresponding pattern recognition.

INSULTING AND ABUSING LANGUAGE DICTIONARY

We have collected approximately 2,700 words, phrases and expressions, with different degrees of manifestation of various flames. All the entries in this dictionary have a considerable load of either *abusing / insulting* impact or *extreme subjectivity* in some of the categories listed above. We initially assigned all the entries weights in the range of 1

⁸⁵ In pre-processing, first all the different headers, internet addresses, email addresses and tags were filtered out. Then all the delimiters, such as spaces, tabs or new line characters, in addition to the following characters: \ \r : () ` 1 2 3 4 5 6 7 8 9 0 \ ' , ; = \ [] ; / < > { } | ~ @ # \$ \% ^ & * _ + were removed from each message, whereas expressive characters (punctuations) such as: - . ' ' ! ? were retained. Punctuations (including “ ”) can be useful for determining the scope of a speaker's messages. This step prevents the system from creating a lot of useless tokens as features for our first-level classifier.

to 5, based on the potential impact level of each entry on the classification of the containing context. These weights can be used for setting the tolerance margin on flame detection for different applications. Then, in several steps of adaptive learning (on training data), we performed modifications on the weights to tailor the task for the most generic purpose; the process of the adaptive learning could be performed based on any targeted specific domain in the field of the flame detection. We achieved stability for the weights with the highest level of discrimination on flames/non-flames. The result is our Insulting or Abusive Language Dictionary (IALD), a fundamental resource for our system.

In the beginning, some of these phrases and expressions contained up to five words, including some wild-cards like 'Somebody' or 'Something' (e.g. "*chew Somebody's ass out*" or "*bail Somebody or Something out*"). These entries are actually raw texts that became patterns in the next stage; they help the software estimate the probability of each context being a flame. At this level, we create a pattern for each of the entries that matches a variety of word sequences (e.g. replacing Somebody or Something wildcards in the example above). In this way, each pattern can be matched with any sequence of words in which we have some (not more than three) tokens in place of wild cards. The patterns could also match series using different types of verbs ending in *ing*, *ed*, *d*, *es* or *s*) or nouns ending in *es*, *s*⁸⁶. Hence, the original patterns in the repository entries were generalized, achieving considerable flexibility; they now could match tens of thousands of word sequences in everyday contexts.

⁸⁶ In addition to matching the wildcards, any word, phrase or expression which has a special character (leading or tailing) in the message would be tested and matched with the corresponding IADL entry.

At this level, after pattern matching for each message/sentence, we could apply another resource for flame probability estimation for the main task, flame detection.

MULTILEVEL CLASSIFICATION

As part of the machine learning core of our package, we run three-level classifications on training data using the IAL Dictionary.

Due to the high degree of feature sparsity, in the first level of classification we use the *Complement Naïve Bayes classifier* [Wit08] to select the most discriminative (~1700) features⁸⁷ as the new training feature space, and pass them to the next level of classification. The initial raw data after tokenization contained 15,636 features after preliminary feature trimming (i.e. the removal of stop-words and terms that occurred only once).

For the second level, we chose the *Multinomial Updatable Naïve Bayes classifier* [Wit08], to efficiently update its model (Model 2) based on new labeled sentences which could be added to the system after the initial training process, in order to carry out adaptive learning. This classifier was run on the best feature space extracted from the previous level of classification. The outputs of this classification level are new aggregated features extracted from the previous level feature space, with the following attributes as input for our last-level classification task, using IALD:

- Frequency of IALD word/phrase/expression patterns that are matched in the current instance, in each weight level (five attributes);
- Maximum weight of IALD entries that are matched in the current message;

⁸⁷ We used the Wrapper Supervised Feature Selection algorithm with "RankSearch" for our search method in Weka.

- Normalized average weight of IALD entries that are matched in the current message;
- The probability that the current instance is *Okay*, based on the previous level classification after applying Model 2;
- The probability that the current instance is a *Flame*, based on the previous level classification after applying Model 2;
- The prediction of the previous level classification on the current instance, after applying Model 2 (*Okay or Flame*);

In the last level, we run a *rule-based classifier* called DTNB (Decision Table/Naive Bayes hybrid classifier [Hal08]) on the output of the second level (i.e. the features described above and label assigned in the previous level), which makes the final decision regarding the current instance: *Okay or Flame*.⁸⁸

RESULTS

After pre-processing, and before performing feature selection, we ran the Complement Naïve Bayes classifier on the whole feature space (15,636). Applying 10-fold cross-validation on the above data gave us the results depicted in the first row of Tables 17 and 18.

At this level, the accuracy was approximately 16% better than the baseline. The baseline that we use for comparison always selects the most frequent class (reflecting the class distribution), and has an accuracy of 68%. As shown in Table 13, there were 936 *Okay* texts classified correctly as *Okay*, and 349 *Flames* classified correctly as *Flames*. The

⁸⁸ As most of the computation is run prior to the final detection, the system could easily be applied in online interactive applications.

others are classification errors: 102 Flames classified as Okay, and 138 Okay texts classified as Flames.

Since the 10-fold cross-validation works on features selected from the entire dataset, it is different from the operation of a deployed package, where the test instances do not participate in the feature selection process. To evaluate the performance more realistically, we separately trained then tested a held-out (10%) randomly selected test file for system stability verification. At the same level, we applied the method on a 10% test set (same baseline), and trained the method based on the rest of the data, achieving the results shown in the second row of Tables 17 and 18.

True Pos.	False Pos.	True. Neg.	False Neg.	Classification#
936	102	349	138	1
89	16	36	11	2
999	39	385	102	3
84	3	27	8	4
1022	16	454	33	5
86	0	32	4	6

Table 17. Flattened confusion matrices for all 6 classification results. True Pos. shows the number of texts which correctly classified as Okay; False Pos. shows the number of texts which falsely classified as Okay; True Neg. shows the number of texts which correctly classified as Flame, and False Neg. shows the number of texts which falsely classified as Flame.

At the second classification level, we used the most expressive selected features (~1,700 features selected by classification); the results of the Naïve Bayes Multinomial Updateable Classifier, applied with 10-folds cross-validation are in the third row of

Tables 17 and 18. These results show that the second level of classification increased the software performance by approximately 7%.

As above, we applied the method on a 10% test set (same baseline), trained the system based on the rest, and achieved the results shown in the fourth row in Tables 13 and 14.

At this stage, increasing the system's discriminative power and going beyond the previous-level accuracy (~91%) was difficult. The software required a lot of consideration, and evaluation of the structural details of the IALD entries, to increase the detection power over 91%. Hence, we applied the DTNB (Decision Table/Naive Bayes hybrid classifier) rule-based classifier, based on extra information extracted from the IALD and its built-in semantic rules (pattern matching modules).

The third level results, using 10-fold cross-validation are in row 5 of the Tables 17 and 18. This result shows that performing the last level significantly improves the accuracy by 6%.

Results →		Correctly	Incorrectly	Okay	Flame	Row
Experiments ↓		Classified	Classified	Precision	Precision	No
First level Classification	10 old C.V.	84.26%	15.73%	87.2%	77.4%	1
	10% Test Size	81.37%	18.62%	86.0%	56.3%	2
Second level Classification	10 Fold C.V.	90.75%	9.24%	90.7%	90.8%	3
	10% Test Size	90.98%	9.01%	9.13%	90.0%	4
Third level Classification	10 Fold C.V.	96.78%	3.21%	96.9%	96.6%	5
	10% Test Size	96.72%	3.27%	95.6%	100%	6

Table 18. Performance comparison along the three levels of classifications, for cross-validation (C.V.) of the training data and the test set.

As with the previous levels, we wanted to verify the stability of the results, so we applied the method on a test set of 10% of the data, and obtained the results shown in row 6 of Tables 17 and 18.

When we evaluated the above results, as well as the results of numerous other experiments we ran, it was clear that the stability of the system increased after each level, and at the last level the results on cross-validation and the test set were quite similar.

If we consider the pair-wise agreement of judges on the data from the previous annotation project [173] (which was part of our data), we find that the pair-wise agreement between human judges (based on the same definition of a flame message) averaged 92%. Other survey results (on similar but different data) showed that, although the agreement rate was 98% for non-flammatory messages, this decreased to 64% consensus for flame messages [173]. One important issue for human annotation which should be taken into account is that the distribution of the data (balanced/unbalanced) has minimal influence on human judgments, unlike the case for the machine learning classifiers.

Our higher percentage of agreement with the labels indicates that the current software has a high level of adaptability, based on the training dataset and the IALD patterns and weights. Therefore, we can conclude that our method has significant capacity for customization for use in specific applications.

Discrepancies between human judges (with the same problem definition) could be due to differences in sensitivity, mood, background and other subjective conditions; being subjective, human judgment is not necessarily the same among different people. Thus, it is helpful to have a standard detection system that can make judgments based on constant predefinitions, patterns and rules.

Unfortunately, no flame detection software is yet available for trial or research purposes, so we cannot directly compare our results to the results of other systems on our dataset.

DISCUSSION

Many of our IALD entries are applied as semantic classification rules. In the third level of classification, we attempt to match each of the corresponding patterns, built with respect to the entry's *wildcards*, on additional prefixes, suffixes or special characters (leading or trailing). This helps to distinguish whether the containing instance is a *Flame* or an *Okay*.

The advantages of the method could be:

The software can be used for message level or sentence level classification tasks in real-time applications (a fraction of a second for each new context).

Our system benefits from both statistical models and rule-based patterns, in addition to specific semantic patterns inside the IALD; it does not rely on only one of these.

Our software is not overly sensitive to punctuation and grammatical mistakes.

The method could be adapted in time, based on user feedback.

One of the limitations of our system is that it does not consider the syntactical structure of the messages explicitly. It could be equipped with modules designed for subjectivity detection based on their lexicons, in which case the length of each message would be a limitation.

As we apply patterns from IALD, and classifier models for flame detection, it is important to prevent training the classifiers based on instances in which the assigned labels are opposed to some of the IALD built-in weighted patterns, and vice versa. Otherwise, the system will suffer from a considerable level of noise in the data.

In future, we will add a synchronized adaptive weight modifier module to the IALD accessory, based on further available training data.

CONCLUSION

We have designed and implemented new and very efficient flame detection software. It applies models from multi-level classifiers, boosted by an Insulting and Abusing Language Dictionary. We built two rule-based auxiliary systems: one is the last level of our classifiers, and the other is used for building patterns out of the IALD repository. The software performs with a high degree of accuracy, for both normal text and flames.

Our flame detection method can be modified based on any accumulative training data, and applied on all collaborative writing websites where people can add or modify content, as in the style of Wikipedia. It could be helpful for web-logs or specialist forums, and be adapted for spam detection on any text messaging services, such as cellular phone SMS. It could also be useful for text chat services, as well as any comment acceptance posts on social networking sites like *Orkut* and *Facebook*.

Bibliography

- [Agg99] Aggarwal C. C., Procopiuc C., Wolf J. L., Yu P. S., and Park J. S.; Fast algorithms for projected clustering. In *ACM SIGMOD International Conference on Management of Data*, 1999.
- [Agg00] Aggarwal, C.C. and Yu, P.S. 2000. Finding generalized projected clusters in high dimensional spaces. In *Proc. Of SIGMOD 2000 Conference*, pp. 70–81.
- [Agr05] Agrawal, Rakesh; Gehrke, Johannes; Gunopulos, Dimitrios; Raghavan, Prabhakar (2005), "Automatic Subspace Clustering of High Dimensional Data", *Data Mining and Knowledge Discovery (Springer Netherlands)* 11 (1): 5–33,
- [Ahm10] Ahmed, M.S., Khan, L., Rajeswari, M.: Using Correlation Based Subspace Clustering for Multi-label Text Data Classification. ;In *ICTAI (2)* (2010) 296-303
- [Ama07] Aman, S. & Szpakowicz. S. (2007). Identifying Expressions of Emotion in Text. *Lecture Notes in Computer Science*. 4629: 196--205.
- [Ami07] Amini M.-R., Usunier N. "A Contextual Query Expansion Approach by Term Clustering for Robust Text Summarization ";*Proceedings of the 7th Document Understanding Conference (DUC 2007)*
- [Arv08] Arvidsson, F. and Flycht-Eriksson, A.: *Ontologies I*. Retrieved 26 Nov 2008.
- [Ban03] Banerjee, S. ; Pedersen, T. : Extended Gloss Overlaps as a Measure of Semantic Relatedness. *IJCAI 2003*: 805-810
- [Bak98] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In *Proceedings of SIGIR'98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96-103, Melbourne, AU, 1998. ACM Press, New York, US.
- [Ban07] Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., & Modha, D. S. (2007). A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8, 1919–1986.
- [Bek01] Bekkerman R., El-Yaniv R., Winter Y. and Tishby N.: On Feature Distributional Clustering for Text Categorization. *SIGIR 2001*: 146-153
- [Bek02] Bekkerman R., El-Yaniv R. , Tishby N. and Winter Y.: Distributional Word Clusters vs. Words for Text Categorization. *Journal of Machine Learning Research* 3: 1183-1208 (2002)
- [Bel57] Bellman, R.E. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ. Republished 2003: Dover, ISBN 0486428095.
- [Ber92] Bergadano, F., Matwin, S., Michalski, R., Zhang, J.: Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System. *Machine Learning* 8(1), 5–43-1992
- [Bin01] Bingham, E., & Mannila, H. (2001). Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the 7th ACM SIGKDD* (pp. 245–250).
- [Ble03] Blei David M.; Andrew Y. Ng, Michael I. Jordan (2003). "Latent Dirichlet Allocation". *Journal of Machine Learning Research* 3: 993–1022. doi:10.1162/jmlr.2003.3.4-5.993

[Boh04] Bohm, Christian; Kailing, Karin; Kriegel, Hans-Peter; Kröger, Peer (2004), "Density Connected Clustering with Local Subspace Preferences", *Data Mining, IEEE International Conference on* (Los Alamitos, CA, USA: IEEE Computer Society): 24–34, ISBN 0-7695-2142-8

[Bre96a] Breiman, L., *Stacked Regression*, *Machine Learning*, 24, 1996

[Bre96b] BREIMAN, Leo, 1996. *Bagging Predictors*. *Machine Learning*, 24(2), 123–140.

[Bru99] Bruce, R.F., Wiebe, J.: *Recognizing subjectivity: a case study in manual tagging*. *Natural Language Engineering* 5 (2), (1999)

[Cai06] Keke Cai, Jiajun Bu, Chun Chen, Peng Huang: *An Automatic Approach for Efficient Text Segmentation*. *KES* (1) 2006: 417-424

[Cam03] F. Camastra. *Data dimensionality estimation methods: a survey*. *Pattern Recognition*, 36(12):2945– 2954, Dec. 2003.

[Can07] CANUTO, Anne M. P., et al., 2007. *Investigating the Influence of the Choice of the Ensemble Members in Accuracy and Diversity of Selection-Based and Fusion-Based Methods for Ensembles*. *Pattern Recognition Letters*, 28(4), 472–486.

[Car84] Carolyn R. Miller; *GENRE AS SOCIAL ACTION-Quarterly journal of speech*, 70 (1984) 151-167

[Cha04] CHAWLA, Nitesh V., et al., 2004. *Learning Ensembles from Bites: A Scalable and Accurate Approach*. *Journal of Machine Learning Research*, 5, 421–451.

[Cha11] Chaffar, S., Inkpen, D. (Inpress). *Towards Emotion Detection in Text.*; *IEEE transactions in Affective Computing* (2011).

[Che96] Chen, SF; Goodman, J; (1996). "An empirical study of smoothing techniques for language modeling". *Proceedings of the 34th annual meeting on Association for Computational Linguistics*.

[CHO95] CHO, Sung-Bae, and Jin H. KIM, 1995. *Multiple Network Fusion Using Fuzzy Logic*. *IEEE Transactions on Neural Networks*, 6(2), 497–501.

[Chu90] Church, K. W., & Hanks, P. (1990). *Word association norms, mutual information and lexicography*. *Computational Linguistics*, 16, 22-29.

[Cla08] CLAESKENS, Gerda, and Nils Lid HJORT, 2008. *Model Selection and Model Averaging*. Volume 27 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge: Cambridge University Press.

[Dav03] Dave, K., Lawrence, S., Pennock, D.M.: *Mining the peanut gallery: Opinion extraction and semantic classification of produce reviews*. In *Proceedings of the 12th International World Wide Web Conference* (2003)

[Dee88] Deerwester, S., et al, *Improving Information Retrieval with Latent Semantic Indexing*, *Proceedings of the 51st Annual Meeting of the American Society for Information Science* 25, 1988, pp. 36–40.

[Dee90] Deerwester, S. C.; Dumais, S. T.; Landauer, T. K.; Furnas, G. W.; Harshman, R. A.: *Indexing by latent semantic analysis*. *Journal of the American Society of Information Science* 41 (6), 391- 407. 1990.

[Del02a] Delin, J. ; Bateman, j. and Allen, P.; *A Model of Genre in Document Layout* ; Source: *Information Design Journal*, Volume 11, Number 1, 2002 , pp. 54-66(13)

- [Del02b] Delorme, M.-A., Lortie-Lussier, M., and De Koninck, J. (2002). *Stress and Coping in the Waking and Dreaming States during an Examination Period*. *Dreaming*, 12(4), pp. 171-183.
- [Dem90] Demmel, J.; Kahan, W. (1990), "Accurate singular values of bidiagonal matrices", *Society for Industrial and Applied Mathematics. Journal on Scientific and Statistical Computing* 11 (5): 873–912, doi:10.1137/0911052 .
- [Dhi01] Dhillon, I. S. (2001). *Co-clustering documents and words using bipartite spectral graph partitioning*. In *Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 269–274).
- [Die00] DIETTERICH, T. G., 2000. *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization*. *Machine Learning*, 40(2), 139–157.
- [Dom03] Domhoff, G. W. (2003). *The Scientific Study of Dreams: Neural Networks, Cognitive Development, and Content Analysis*. American Psychological Association, New York.
- [Don00] Donoho D. L.; High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality. In *Mathematical Challenges of the 21st Century*, University of California Los Angeles, Aug. 2000.
- [Dum05] Dumais S. T. (2005). *Latent Semantic Analysis*. doi:10.1002/aris.1440380105.
- [Ekm78] Ekman, P. & Friesen, W. V. (1978) *Facial action coding system: Investigator's guide*, Palo Alto, Consulting Psychologists Press.
- [Eme04] Emerson, C.; Bakhtin, M.M.; Holquist, M : *Speech Genres and Other Late Essays*; Edition: 9 – 2004
- [Est96] Ester M., Kriegel H.-P., Sander J. and Xu X. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*. Portland, OR, 226-231.
- [Fir57] Firth J.R. et al. *Studies in Linguistic Analysis. A synopsis of linguistic theory, 1930-1955. Special volume of the Philological Society*. Oxford: Blackwell, 1957.
- [Fra07] D. Francois, V. Wertz, and M. Verleysen. *The Concentration of Fractional Distances*. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):873–886, July 2007.
- [Fre05] A. L. Fred and A. K. Jain. *Combining Multiple Clusterings Using Evidence Accumulation*. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(6):835–850, June 2005.
- [Fre08] Fredrik Arvidsson and Annika Flycht-Eriksson. *Ontologies I*. Retrieved 26 Nov 2008.
- [Fre96] FREUND, Yoav, and Robert E. SCHAPIRE, 1996. *Experiments with a New Boosting Algorithm*. In: Lorenza SAITTA, ed. *Machine Learning: Proceedings of the Thirteenth International Conference (ICML '96)*. San Francisco, CA: Morgan Kaufmann, pp. 148–156.
- [Fum05] FUMERA, Giorgio, and Fabio ROLI, 2005. *A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6), 942–956.

- [Gar05] GARCÍA-PEDRAJAS, Nicolás, César HERVÁS-MARTNEZ, and Domingo ORTIZ-BOYER, 2005. *Cooperative Coevolution of Artificial Neural Network Ensembles for Pattern Classification*. *IEEE Transactions on Evolutionary Computation*, 9(3), 271–302.
- [Gor03] Gordon, A., Kazemzadeh, A., Nair, A., Petrova, M.: Recognizing expressions of commonsense psychology in English text. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pp. 208–215 (2003)
- [Gre94] Grefenstette, G. (1994). *Explorations in automatic thesaurus discovery*. Boston: Kluwer.
- [Gri04] Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101, 5228–5235.
- [Gru93] Gruber T., (1993). "A translation approach to portable ontology specifications". In: *Knowledge Acquisition*. 5: 199-199.
- [Hal08] Hall, M., Frank, E.: *Combining Naive Bayes and Decision Tables*. *FLAIRS Conference*: 318-319 (2008)
- [Hal66] Hall, C. S. and Van de Castle, R. L. (1966). *The Content Analysis of Dreams*. Meredith Publishing Company, New York.
- [Han07] HANSEN, Bruce E., 2007. *Least Squares Model Averaging*. *Econometrica*, 75(4), 1175–1189.
- [Har54] Harris, Zellig (1954). "Distributional Structure". *Word* 10 (2/3): 146–62.
- [Har64] Harris, Z.: *Distributional structure*. In Katz, J.J., Fodor, J.A., eds.: *The Philosophy of Linguistics*, New York, Oxford University Press, 1964.
- [Har85] Harris, Z. : *Distributional structure*. In Katz, J.J., ed.: *The Philosophy of Linguistics*. Oxford University Press 26–47, 1985.
- [Har98] Hartmann, E. (1998). *Dreams and Nightmares: The New Theory on the Origin and Meaning of Dreams*. Plenum, New York
- [Hec91] R. HECHT-NIELSEN, *Neurocomputing*, Addison–Wesley Publishing Company, Reading, MA, u.a., 1991.
- [Hir01] Hirano, S.; Inuiguchi, M. and Tsumoto, S. (Eds.), *Proceedings of International Workshop on Rough Set Theory and Granular Computing, a special issue of the Bulletin of International Rough Set Society*, Vol. 5, No. 1-2, 2001.
- [HO94] Ho, Tin Kam, Jonathan J. HULL, and Sargur N. SRIHARI, 1994. *Decision Combination in Multiple Classifier Systems*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1), 66–75.
- [HO98] Ho, Tin Kam, 1998. *The Random Subspace Method for Constructing Decision Forests*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), 832–844.
- [Hob98] Hobson, J. A., Stickgold, R. and Pace-Schott, E. F. (1998). *The Neuropsychology of REM Sleep Dreaming*. *NeuroReport*, Vol. 9, pp. R1-R14.
- [Hoe99] Hoeting, Jennifer A., et al., 1999. *Bayesian Model Averaging: A Tutorial*. *Statistical Science*, 14(4), 382–401.

- [Hof99] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR* (pp. 50–57).
- [Hua08] Huang, A., Milne, D., Frank, E., Witten, I. H.: Clustering Documents with Active Learning using Wikipedia. In *Proceedings of ICDM, IEEE*, pp. 839–844 (2008)
- [Hua09] Huang, A., Milne, D., Frank, E. and Witten, I.H. (2009) Clustering documents using a Wikipedia-based concept representation. In *Proceedings of the Thirteenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09)*, Bangkok, Thailand.
- [Hul94] Hull, D.; *Improving text retrieval for the routing problem using Latent Semantic Indexing*, in *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference*, 1994.
- [Hus31] Husserl, Edmund. *Ideas: General Introduction to Pure Phenomenology*. Translated by W. R. Boyce Gibson. London: George Allen & Unwin Ltd., 1931.
- [Jai00] JAIN, Anil K., Robert P. W. DUIN, and Jianchang MAO, 2000. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 4–37.
- [Jai05] A. K. Jain and M. H. Law. Data Clustering: A User's Dilemma. In S. P. et al., editor, *Proceedings of International Conference on Pattern recognition and machine intelligence (PReMI)*, volume 3776 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, 2005.
- [Kai04] Kailing, Karin; Kriegel, Hans-Peter; Kröger, Peer (2004), "Density-Connected Subspace Clustering for High-Dimensional Data", *Proceedings of the Fourth SIAM International Conference on Data Mining (SIAM)*: 246–257,
- [Kau00] Kaufer, D.: *Flaming: A White Paper*. (2000)
- [Kim08] Kim Y., Pessiot J.-F., Amini M.-R., Gallinari P. "An Extension of PLSA for Document Clustering"; *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008)*, p. 1345-1346
- [Kit03] KITTLER, J., and F. M. ALKOOT, 2003. Sum versus Vote Fusion in Multiple Classifier Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1), 110–115.
- [Kit98a] KITTLER, J., 1998. Combining Classifiers: A Theoretical Framework. *Pattern Analysis and Applications*, 1(1), 18–27.
- [Kit98b] KITTLER, Josef, et al., 1998. On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3), 226–239.
- [Kle00] KLEINBERG, Eugene M., 2000. On the Algorithmic Implementation of Stochastic Discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5), 473–490.
- [Kle90] KLEINBERG, E. M., 1990. Stochastic Discrimination. *Annals of Mathematics and Artificial Intelligence*, 1(1–4), 207–239.
- [Kop00] M. K"oppen. *The Curse of Dimensionality*. In *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, Sept. 2000.

- [Kou09] Kouznetsov, A.; Matwin, S.; Inkpen, D.; Razavi, A. H.; Frunza, O.; Sehatkar, M. and Seaward, L.: "Classifying Biomedical Abstracts Using Committees of Classifiers and Collective Ranking Techniques", The 22th Canadian AI, May 2009.
- [Kri05] Kriegel, Hans-Peter; Kröger, Peer; Renz, Matthias; Wurst, Sebastian (2005), "A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data", *Proceedings of the Fifth IEEE International Conference on Data Mining* (Washington, DC: IEEE Computer Society): 205–257, ISBN 0-7695-2278-5
- [Kri09] Kriegel, H.-P.; Kröger, P.; Zimek, A. (March 2009). "Clustering High Dimensional Data: A Survey on Subspace Clustering, Pattern-based Clustering, and Correlation Clustering". *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3 (1): 1–58.
- [Kro95] KROGH, Anders, and Jesper VEDELSBY, 1995. *Neural Network Ensembles, Cross Validation, and Active Learning*. In: Gerald TESAURO, David S. TOURETZKY, and Todd K. LEEN, eds. *Advances in Neural Information Processing Systems 7*. Cambridge, MA: The MIT Press, pp. 231–238.
- [Kul05a] Kulkarni, A.; Pedersen, T. : Name Discrimination and Email Clustering using Unsupervised Clustering and Labeling of Similar Contexts. *IICAI 2005*: 703-722
- [Kul05b] Kulkarni, A. and Pedersen, T. : *SenseClusters: Unsupervised Clustering and Labeling of Similar Contexts* - Appears in the *Proceedings of the Demonstration and Interactive Poster Session of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 105-108, June 26, 2005, Ann Arbor, MI. [acceptance rate 55%] Download the data used in this paper.
- [Kun00] KUNCHEVA, L. I., et al., 2000. Is Independence Good For Combining Classifiers In: A. SANFELIU, et al., eds. *Proceedings, 15th International Conference on Pattern Recognition, Volume 2*. Los Alamitos: IEEE Computer Society, pp. 168–171.
- [Kun02] KUNCHEVA, Ludmila I., 2002a. A Theoretical Study on Six Classifier Fusion Strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2), 281–286.
- [Kun03a] KUNCHEVA, L. I., et al., 2003. Limits on the Majority Vote Accuracy in Classifier Fusion. *Pattern Analysis and Applications*, 6(1), 22–31. LAM, Louisa, and Ching Y. SUEN, 1995. Optimal Combination of Pattern Classifiers. *Pattern Recognition Letters*, 16(9), 945–954.
- [Kun03b] KUNCHEVA, Ludmila I., and Christopher J. WHITAKER, 2003. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, 51(2), 181–207.
- [Kun07] KUNCHEVA, Ludmila I., and Juan J. RODRÍGUEZ, 2007. Classifier Ensembles with a Random Linear Oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4), 500–508.
- [LAM95] LAM, Louisa, and Ching Y. SUEN, 1995. Optimal Combination of Pattern Classifiers. *Pattern Recognition Letters*, 16(9), 945–954.
- [LAM97] LAM, Louisa, and Ching Y. SUEN, 1997. Application of Majority Voting to Pattern Recognition: An Analysis of Its Behavior and Performance. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 27(5), 553–568.
- [Lan97] Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104, 211-240.

- [Lap05] Lapata, M. and Barzilay, R.: 2005. Automatic Evaluation of Text Coherence: Models and Representations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 1085-1090. Edinburgh.
- [Lar97] LARKEY, Leah S., and W. Bruce Croft, 1997. Combining Classifiers in Text Categorization. In: Hans-Peter Frei, et al., eds. *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM Press, pp. 289–297.
- [Lea08] LEAP, Nathan J., et al., 2008. An Investigation of the Effects of Correlation and Autocorrelation on Classifier Fusion and Optimal Classifier Ensembles. *International Journal of General Systems*, 37(4), 475–498.
- [Lev05] E. Levina and P. J. Bickel. Maximum Likelihood Estimation of Intrinsic Dimension. In L. B. L. K. Saul, Y. Weiss, editor, *Advances in NIPS*, volume 17, 2005. There is a correction on <http://www.inference.phy.cam.ac.uk/mackay/dimension/>.
- [Li 11] Yan Li, Edward Hung, Korris Chung. A subspace decision cluster classier for text classication. *Expert Systems with Applications*, 2011, 38 (10): 12475 - 12482.
- [Lin98] Lin, T.Y.; Granular computing on binary relations I: data mining and neighbourhood systems, in: L. Polkowski and A. Skowron (Eds.), *Rough Sets in Knowledge Discovery 1, Methodology and Applications*, Physica-Verlag, Heidelberg, pp. 286-318, 1998.
- [Lug04] Luger, George; Stubblefield, William (2004). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving* (5th ed.). The Benjamin/Cummings Publishing Company, Inc.. ISBN 0-8053-4780-1. <http://www.cs.unm.edu/~luger/aiinal/tocfull.html>., pp. 227-243
- [Lun95] K. Lund, C. Burgess, R. Atchley; Semantic and associative priming in high-dimensional semantic space; *Proceedings of the 17th annual conference of the Cognitive Science Society*, Erlbaum, Mahwah, NJ (1995), pp. 660–665
- [Lun96] Lund K and Burgess C. Producing high-dimensional semantic spaces from lexical co-occurrence. In *Behavior Research Methods, Instrumentation, and Computers*, pages 203-208, 1996.
- [Mah08] Mahmud, A., Ahmed, K.Z., Khan, M.: Detecting flames and insults in text, *Proc. of 6th International Conference on Natural Language Processing (ICON-2008)*, CDAC Pune, India, December 20 - 22 (2008)
- [Mah36] Mahalanobis, P C (1936). "On the generalised distance in statistics". *Proceedings of the National Institute of Sciences of India* 2 (1): 49–55.
- [Man98] Manning, C. D. and Schütze, H.; *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts. 1998.
- [Maq96] Maquet, P., Péters, J. M., Aerts, J., Delfiore, G., Degueldre, C, Luxen, A. and Franck, G. (1996). Functional Neuroanatomy of Human Rapid-Eye- Movement Sleep and Dreaming, *Nature*, Vol. 383, pp.163-166
- [Mar02] Martin, M.J.: Annotating flames in Usenet newsgroups: a corpus study. For NSF Minority Institution Infrastructure Grant Site Visit to NMSU CS department (2002)
- [Mar07] Marko Robnik-Sikonja, Igor Kononenko: An adaptation of Relief for attribute estimation in regression. In: *Fourteenth International Conference on Machine Learning*, 296-304, 1997.

- [Mas02a] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: *WonderWeb deliverable d17. The WonderWeb library of foundational ontologies and the DOLCE ontology*; (ISTC-CNR-ver. 2.0, 15-08-2002)
- [Mas02b] Masolo, C.; Borgo, S.; Gangemi, A.; Guarino, N. and Oltramari, A.: *Wonderweb deliverable d18:Ontology Library(Final); The WonderWeb library of foundational ontologies*, ISTC-CNR; 2002
- [Mat10] Matwin S., Razavi A. H., De Koninck J., Amini R.R.; *Classification of Dreams Using Machine Learning- ECAI 2010 - European Conference on Artificial Intelligence - PAIS 2010*
- [MCD02] McDonald, S.; Ramscar, M.: *Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In Proceedings of the 23rd Annual Conference of the Cognitive Science Society, 2001.*
- [McE00a] McEnery, A.M., Baker, J.P.,Hardie, A.: 'Swearing and abuse in modern British English', in B. Lewandowska-Tomaszczyk and P.J. Melia (eds) *Practical Applications of Language Corpora*, Peter Lang, Hamburg, pp 37-48. (2000)
- [McE00b] McEnery, A.M., Baker, J.P.,Hardie, J.: 'Assessing claims about language use with corpus data – swearing and abuse', in J. Kirk (ed) *Corpora Galore*, Rodopi, Amsterdam, pp 45-55. (2000)
- [McE04] McEnery, A.M., Xiao, Z.: 'Swearing in modern British English: the case of fuck in the BNC', *Language and Literature*, Volume 13, Issue 3, pp 235-268. ISSN 0963-9470- (2004)
- [McE08] McEnery, A.M.: *Swearing in English: Bad Language, Purity and Power from 1586 to the Present*, Routledge, London. (in press) ISBN 978-0415258371(2005)
- [Mel05] MELVILLE, Prem, and Raymond J. MOONEY, 2005. *Creating Diversity in Ensembles Using Artificial Data. Information Fusion*, 6(1), 99–111.
- [Mih06] Mihalcea R., Corley C., Strapparava C.: *Corpus-based and Knowledge-based Measures of Text Semantic Similarity. AAAI 2006: 775-780*
- [Mil91] Miller, G. and Charles, W. "Contextual correlates of semantic similarity". *Language and Cognitive Processes* 6(1):1–28. 1991.
- [Mil99] MILLER, David J., and Lian YAN, 1999. *Critic-Driven Ensemble Classification. IEEE Transactions on Signal Processing*, 47(10), 2833–2844.
- [Nad06] Nadeau, D., Sabourin, C., De Koninck, J., Matwin, S., Turney, P.D. (2006), *Automatic dream sentiment analysis, Proceedings of the Workshop on Computational Aesthetics at the Twenty-First National Conference on Artificial Intelligence (AAAI-06), Boston, USA*
- [Ng02] Ng, A., Jordan, M., and Weiss, Y. (2002). *On spectral clustering: analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14 (pp. 849 – 856). MIT Press.*
- [Nie05] Nielsen, T. A. and Stenstrom, P. (2005). *What are the Memory Sources of Dreaming? Nature*, Vol. 437, pp.1286-1289.
- [Nil03] Niles, I., Pease, A.: *Towards a standard upper ontology. In: FOIS 2001. Proc. Of the 2nd Inter. Conf. on Formal Ontology in Information Systems (2001)*

- [Nil98] Nilsson, Nils (1998). *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers. ISBN 978-1-55860-467-4., chpt. 18
- [Ong05] St-Onge, M., Lortie-Lussier, M., Mercier, P., Grenier, J. and De Koninck, J. (2005). *Emotions in the Diary and REM Dreams of Young and Late Adulthood Women and Their Relation to Life Satisfaction. Dreaming*, Vol. 15, pp.116-128.
- [Opi99a] OPITZ, David W., 1999. *Feature Selection for Ensembles*. In: *American Association for Artificial Intelligence (AAAI), ed. AAAI-99: Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press, pp. 379–384.
- [Opi99b] OPITZ, David, and Richard MACLIN, 1999. *Popular Ensemble Methods: An Empirical Study*. *Journal of Artificial Intelligence Research*, 11, 169–198.
- [Pan02] Pang, B., Lee, L., Vaithyanathan, SH.: *Thumbs up? Sentiment classification using machine learning techniques*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79–86 (2002)
- [Pan03] Pan, X.; Assal, H.; *Providing context for free text interpretation; 2003 Page(s):704 – 709*
- [Pan05] Abhinay Pandya, Pushpak Bhattacharyya: *Text Similarity Measurement Using Concept Representation of Texts*. *PREMI 2005*: 678-683
- [Pen07] Peng J., Yang D., Tang Sh., Gao J., Zhang P. and Fu Y.: *A Concept Similarity Based Text Classification Algorithm*. *FSKD (1) 2007*: 535-539
- [Par04] Lance Parsons , Ehtesham Haque , Huan Liu, *Subspace clustering for high dimensional data: a review*, *ACM SIGKDD Explorations Newsletter*, v.6 n.1, p.90-105, June 2004
- [Pat03] Patwardhan,S.; Banerjee,S.; Pedersen,T.: *Using Measures of Semantic Relatedness for Word Sense Disambiguation*. *CICLing 2003*: 241-257
- [Paw98] Pawlak, Z. : *Granularity of knowledge, indiscernibility and rough sets*, *Proceedings of 1998 IEEE International Conference on Fuzzy Systems*, pp. 106-110, 1998.
- [Ped01] Pedrycz, W. : *Granular Computing: An Emerging Paradigm*, Springer-Verlag, Berlin, 2001.
- [Ped05a] Pedersen, T.; Purandare, A. and Kulkarni, A.: *Name discrimination by clustering similar contexts*. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 220–231, Mexico City, February 2005.
- [Ped05b] Pedersen,T.; Kulkarni,A,: *Identifying Similar Words and Contexts in Natural Language with SenseClusters*. *AAAI 2005*: 1694-1695
- [Ped06a] Pedersen, T. and Kulkarni, A.: *Selecting the right number of senses based on clustering criterion functions*. In *Proceedings of the Posters and Demo Program of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 111–114, Trento, Italy, April 2006.
- [Ped06b] Pedersen, T.; Kulkarni, A.; Angheluta, R.; Kozareva, Z. and Solorio, T.: *An unsupervised language independent method of name discrimination using second order co-occurrence features*. In *Proceedings of the Seventh International Conference on Intelligent Text Processing and Computational Linguistics*, pages 208–222, Mexico City, February 2006.

- [Ped07] Pedersen, T.; Kulkarni, A.: *Unsupervised discrimination of person names in web contexts*. In: *CICLing*. 299-310, 2007.
- [Ped97a] Pedersen, T. and Bruce, R.: *Distinguishing word senses in untagged text*. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, Providence, RI, August 1997.
- [Ped97b] Pedersen, T. : *Naive Mixes for Word Sense Disambiguation*. AAAI/IAAI 1997: 841
- [Ped98a] Pedersen, T. and Bruce, R.: *Knowledge lean word sense disambiguation*. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 800–805, Madison, WI, July. 1998.
- [Ped98b] Pedersen, T.: *Raw Corpus Word Sense Disambiguation*. AAAI/IAAI 1998: 1198
- [Pen01] Pennebaker, J. W., Francis, M. E. and Booth, R.J. (2001) *Linguistic Inquiry and Word Count LIWC2001*. Erlbaum Publishers, Mahwah, NJ.
- [Per93a] Perrone, Michael P., and Leon N. COOPER, 1993. *When Networks Disagree: Ensemble Methods for Hybrid Neural Networks*. In: R. J. MAMMONE, ed. *Neural Networks for Speech and Image Processing*. London: Chapman-Hall, pp. 126–142.
- [Per93b] Pereira, F. , Tishby N. and Lee L. “*Distributional clustering of English words*”; In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183-90, 1993.
- [Pes10] Pessiot J.-F., Kim Y., Amini M.-R., Gallinari P.” *Improving Document Clustering in a Learned Concept Space*” ; *Information Processing & Management Elsevier*, 46(2):180-192, 2010
- [Pet79] K. W. Pettis, T. A. Bailey, A. K. Jain, and R. C. Dubes. *An Intrinsic Dimensionality Estimator from Near-Neighbor Information*. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-1(1):25–37, Jan. 1979.
- [Pol98] Polkowski, L. and Skowron, A. *Towards adaptive calculus of granules*, *Proceedings of 1998 IEEE International Conference on Fuzzy Systems*, pp. 111-116, 1998.
- [Poo98] Poole, David; Mackworth, Alan; Goebel, Randy (1998). *Computational Intelligence: A Logical Approach*. New York: Oxford University Press. <http://www.cs.ubc.ca/spider/poole/ci.html>., pp. 23-46, 69-81, 169-196, 235-277, 281-298, 319-345
- [Pos68] Posner, M. I., & Keele, S. W. (1968). *On the genesis of abstract ideas*. *Journal of Experimental Psychology*, 77, 353-363.
- [Pur04a] Purandare, A.; Pedersen, T.: *SenseClusters - Finding Clusters that Represent Word Senses*. AAAI 2004: 1030-1031
- [Pur04b] Purandare, A. and Pedersen, T.: *Word sense discrimination by clustering contexts in vector and similarity spaces*. In *Proceedings of the Conference on Computational Natural Language Learning*, Boston, MA. 2004.
- [Raf97] RAFTERY, Adrian E., David MADIGAN, and Jennifer A. HOETING, 1997. *Bayesian Model Averaging for Linear Regression Models*. *Journal of the American Statistical Association*, 92(437), 179–191.
- [Raz10] Razavi A.H., Inkpen D., Uritsky S., Matwin, S.: *Offensive Language Detection Using Multi-level Classification*. *Canadian Conference on AI 2010*: 16-27

[Raz08a] Razavi, A.H., Amini, R., Sabourin, C., Sayyad Shirabad, J., Nadeau, D., Matwin, S., De Koninck, J.: *Classification of emotional tone of dreams using machine learning and text analyses*. (Paper presented at the Meeting of the Associated Professional Sleep Society in Baltimore. *Sleep*, 31, A380-381 (2008)

[Raz08b] Razavi, A.H., Amini, R., Sabourin, C., Sayyad Shirabad, J., Nadeau, D., Matwin, S., De Koninck, J.: *Evaluation and Time Course Representation of the Emotional Tone of dreams Using Machine Learning and Automatic Text Analyses*. - 19th Congress of European Sleep Research Society; *ESRS-Glasgow Journal of Sleep Research*. (2008)

[Raz09] Razavi A.H., Matwin S., Inkpen D., Kouznetsov A.: *Parameterized Contrast in Second Order Soft Co-occurrences: A Novel Text Representation Technique in Text Mining and Knowledge Extraction*. *ICDM Workshops 2009*: 471-476

[Rey06] REYZIN, Lev, and Robert E. SCHAPIRE, 2006. *How Boosting the Margin can also Boost Classifier Complexity*. In: *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*. New York: ACM, pp. 753– 760.

[Ric91] Richard A. Spears. *Forbidden American English*, ISBN: 9780844251493; (1991)

[Ril03a] Riloff, E., Wiebe, J., Wilson, T.: *Learning subjective nouns using extraction pattern bootstrapping*. In *Proceedings of the 7th Conference on Natural Language Learning*, pp. 25–32. (CoNLL), (2003)

[Ril03b] Riloff, E., Wiebe, J.: *Learning extraction patterns for subjective expressions*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, pp. 105–112 (2003)

[Ros00] Rosario, B.: *Latent Semantic Indexing: An overview*; *INFOSYS 240 Spring 2000-Final Paper*

[Ros78] Rosch, E. (1978). *Principles of categorization*. In E. Rosch & B. B. Loyd (Eds.), *Cognition and categorization* (pp. 28-71). Hillsdale, NJ.: Erlbaum.

[Rus03] Russell, Stuart J.; Norvig, Peter (2003), *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-790395-2, pp. 320-328

[Rut05] RUTA, Dymitr, and Bogdan GABRYS, 2005. *Classifier Selection for Majority Voting*. *Information Fusion*, 6(1), 63–81.

[Sah04] Sahlgren, M., Coster, R.: *Using bag-of-concepts to improve the performance of support vector machines in text categorization*. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING'04* (pp.487-493) (2004)

[Sal83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[Sal88] G. Salton and C. Buckley. *Term-weighting approaches in automatic text retrieval*. *Information Processing and Management*, 24(5):513–523, 1988.

[Sch92] Schutze, H. (1992). *Context space*. In *Fall Symposium on probability and natural language*. Cambridge, MA.: American Association for Artificial Intelligence.

[Sch94] Scherer, K. R. & Wallbott, H.G. (1994). *Evidence for universality and cultural variation of differential emotion response patterning*. *Journal of Personality and Social Psychology*, 66:310-328.

- [Sch95] Schitze, H. & Pedersen, J. O. (1995). *Information retrieval based on word senses*. Fourth Annual Symposium on Document Analysis and Information Retrieval, 161 - 175.
- [Sch97] Scherer, K. R. (1997). *The role of culture in emotion-antecedent appraisal*. *Journal of Personality and Social Psychology*, 73, 902-922.
- [Sch98b] SCHAPIRE, Robert E., et al., 1998. *Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods*. *The Annals of Statistics*, 26(5), 1651–1686.
- [Sch98a] Schütze, H.. *Automatic word sense discrimination*. *Computational Linguistics*, 24(1):97–123. 1998.
- [Sch99] SCHAPIRE, Robert E., 1999. *A Brief Introduction to Boosting*. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, pp. 1401–1406.
- [Sew08] Sewell M., *Ensemble Learning* (2008)
- [Shi00] Shi, J. and Malik, J. (2000). *Normalized cuts and image segmentation*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8), 888 – 905.
- [shi02] SHIPP, Catherine A., and Ludmila I. KUNCHEVA, 2002. *Relationships Between Combination Methods and Measures of Diversity in Combining Classifiers*. *Information Fusion*, 3(2), 135–148.
- [Sim87] Simons, P. (1987) *Parts: a Study in Ontology*. Clarendon Press, Oxford.
- [Sko98] Skowron, A. and Stepaniuk, J.: *Information granules and approximation spaces*, *Proceedings of Seventh International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 354-361, 1998.
- [Sku01] SKURICHINA, Marina, 2001. *Stabilizing Weak Classifiers: Regularization and Combining Techniques in Discriminant Analysis*. Ph. D. thesis, Delft University of Technology, Delft.
- [Sku02] SKURICHINA, Marina, and Robert P. W. DUIN, 2002. *Bagging, Boosting and the Random Subspace Method for Linear Classifiers*. *Pattern Analysis and Applications*, 5(2), 121–135.
- [Slo01] Slonim, N., Tishby, N.: *The Power of Word Clusters for Text Classification*. In: *23rd European Colloquium on Information Retrieval Research* (2001)
- [Smi81] Smith, E. E., & Medin, D. L. *Categories and concepts*. Cambridge, MA: Harvard University Press. (1981).
- [Smy99] Smyth, P. and Wolpert, D. H., *Linearly Combining Density Estimators via Stacking*, *Machine Learning Journal*, 36, 59-83, 1999
- [Sol96] SOLLICH, Peter, and Anders KROGH, 1996. *Learning with Ensembles: How Over-Fitting can be Useful*. In: David S. TOURETZKY, Michael C. MOZER, and Michael E. HASSELMO, eds. *Advances in Neural Information Processing Systems 8*, Bradford Books. Cambridge, MA: The MIT Press, pp. 190–196.
- [Spa72] Spark Jones, K.; "A statistical interpretation of term specificity and its application in retrieval". *Journal of Documentation* 28 (1): 11–21, 1972.
- [Spe97] Spertus, E. *Smokey: Automatic recognition of hostile messages*. In *Proceedings of the Eighth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pp. 1058-1065 (1997)

- [Sus05] Susan T. Dumais (2005). "Latent Semantic Analysis". *Annual Review of Information Science and Technology* 38: 188. doi:10.1002/aris.1440380105
- [Tac05] Täckström, Oscar (2005) *An Evaluation of Bag-of-Concepts Representations in Automatic Text Classification*. Masters thesis, KTH.
- [TAX00] TAX, David M. J., et al., 2000. Combining Multiple Classifiers by Averaging or by Multiplying? *Pattern Recognition*, 33(9), 1475–1485.
- [The08] Thelwall, M.: *Fk yea I swear: Cursing and gender in a corpus of MySpace pages*, *Corpora*, 3(1), 83-107 (2008)
- [Tish99] N. Tishby, F.C. Pereira and W. Bialek. *The Information Bottleneck Method* In *Proc. of the 37-th Allerton Conference on Communication and Computation*, 1999.
- [Tre97] Trefethen, Lloyd N.; Bau III, David (1997), *Numerical linear algebra*, Philadelphia: Society for Industrial and Applied Mathematics, ISBN 978-0-89871-361-9 .
- [Tur02] Turney, P. (2002) *Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews*. *Proc. 40th Annual Meeting of the Association for Computational Linguistics*.
- [Tur03] Turney, P., Littman, M.: *Measuring praise and criticism: Inference of semantic orientation from association*. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346, (2003)
- [Tur08] Turney, P.D. (2008), *The latent relation mapping engine: Algorithm and experiments*, *Journal of Artificial Intelligence Research (JAIR)*, 33, 615-655. (NRC #50738)
- [Tur10] Turney, P.D., and Pantel, P. (2010), *From frequency to meaning: Vector space models of semantics*, *Journal of Artificial Intelligence Research (JAIR)*, 37, 141-188.
- [Tur11] Turney, P.D., Neuman, Y., Assaf, D., and Cohen, Y. (2011), *Literal and metaphorical sense identification through concrete and abstract context*, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-2011)*, Edinburgh, Scotland, UK, July 2011, pp. 680-690.
- [Usc04] Uschold, M., Gruninger, M. (2004) "Ontologies and Semantics for Seamless Connectivity", *SIGMOD Rec.*, 33(4):58-64, 2004.
- [Val04] VALENTINI, Giorgio, and Thomas G. DIETTERICH, 2004. *Bias-Variance Analysis of Support Vector Machines for the Development of SVM-Based Ensemble Methods*. *Journal of Machine Learning Research*, 5, 725–775.
- [Val08] Vale, M, *A Look at High Dimensional Space*; Swiss National Supercomputing Centre (CSCS), September 12, 2008
- [Wie01] Wiebe, J., Bruce, R., Bell, M., Martin, M., Wilson, T.: *A Corpus Study of Evaluative and Speculative Language*. *Proceedings of 2nd ACL SIGdial Workshop on Discourse and Dialogue*. Aalborg, Denmark (2001)
- [Wie04] Wiebe, J., Wilson, T., Bruce, R. Bell, M. and Martin, M.: *Learning Subjective Language*. *Computational Linguistics*, 30, (3):277-308 (2004)
- [Wie05] Wiebe, J., Wilson, T., Cardie, C.: *Annotating expressions of opinions and emotions in language*. *Language Resources and Evaluation*, 39 (2-3): 165-210 (2005)

- [Wie99] Wiebe, J., Bruce, R.F., O'Hara, T.: Development and use of a gold standard data set for subjectivity classifications. In *Proc. 37th Annual Meeting of the Assoc. for Computational Linguistics (ACL-99)*, pp. 246-253 (1999)
- [Wit05] Witten, I. H. and Frank, E. (2005) *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco.
- [Wit08] Witten, I., Frank, E., Gray, J.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, ISBN13: 9781558605527, (2008)
- [Wol92] WOLPERT, David H., 1992. Stacked Generalization. *Neural Networks*, 5(2), 241–259.
- [Xu03] Xu, W., Liu, X., & Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR* (pp. 267–273).
- [XU92] XU, Lei, Adam KRZYŻAK, and Ching Y. SUEN, 1992. Methods of Combining Multiple Classifiers and their Applications to Handwriting Recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3), 418–435.
- [Yag98] Yager, R.R. and Filev, D.: Operations for granular computing: mixing words with numbers, *Proceedings of 1998 IEEE International Conference on Fuzzy Systems*, pp. 123-128, 1998.
- [Yao00] Yao, Y.Y.: Information tables with neighborhood semantics, *Data Mining and Knowledge Discovery: Theory, Tools, and Technology II*, B.V. Dasarathy (Ed.), The International Society for Optical Engineering, Bellingham, Washington, pp. 108-116, 2000.
- [Yao01] Yao, Y.Y.: Modeling data mining with granular computing, *Proceedings of the 25th Annual International Computer Software and Applications Conference*, pp. 638-643, 2001.
- [Yao02] Yao, Y. Y.; Churn-jung Liau, A generalized decision logic language for granular computing, ISBN: 0-7803-7280-8 (2002)
- [Yi03] Yi, J., Nasukawa, T., Bunescu, R., Niblack, W.: Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM-2003)* (2003)
- [Yu03] Yu, H., Hatzivassiloglou, V.: Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 129–136 (2003)
- [Zad79] Zadeh, L.A. : Fuzzy sets and information granularity, in: *Advances in Fuzzy Set Theory and Applications*, Gupta, N., Ragade, R. and Yager, R. (Eds.), North-Holland, Amsterdam, pp. 3-18, 1979.
- [Zad97] Zadeh, L.A. : Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, *Fuzzy Sets and Systems*, 19, 111-127, 1997.
- [Zha08] ZHANG, Chun-Xia, and Jiang-She ZHANG, 2008. A Local Boosting Algorithm for Solving Classification Problems. *Computational Statistics & Data Analysis*, 52(4), 1928–1941.
- [Zho99] Zhong, N.; Skowron, A. and Ohsuga S. (Eds.), *New Directions in Rough Sets, Data Mining, and Granular-Soft Computing*, Springer-Verlag, Berlin, 1999.

[Zim08] Zimek, A. (2008). *Correlation Clustering Flexible Concepts: The POSEIDON System*. *Machine Learning* 8(1), 5–43-1992