# Integrating Structure and Meaning

## Using Holographic Reduced Representations to Improve Automatic Text Classification

by

Jonathan M. Fishbein

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2008

## Abstract

Current representation schemes for automatic text classification treat documents as syntactically unstructured collections of words (Bag-of-Words) or 'concepts' (Bag-of-Concepts). Past attempts to encode syntactic structure have treated part-of-speech information as another word-like feature, but have been shown to be less effective than non-structural approaches. We propose a new representation scheme using Holographic Reduced Representations (HRRs) as a technique to encode both semantic and syntactic structure, though in very different ways. This method is unique in the literature in that it encodes the structure across all features of the document vector while preserving text semantics. Our method does not increase the dimensionality of the document vectors, allowing for efficient computation and storage. We present the results of various Support Vector Machine classification experiments that demonstrate the superiority of this method over Bag-of-Concepts representations and improvement over Bag-of-Words in certain classification contexts.

**Acknowledgements**

I would like to thank all the following people for their help and guidance throughout my degree:

Chris Eliasmith for supervision, guidance, patience, inspiration and encouragement;

Oscar Täckström for guidance and support at the early stages of this research, as well as teaching me how to structure and write a thesis;

Bryan Tripp for teaching me how to be a successful graduate student and CNRG lab member, as well as providing a sounding board for some of my crazy research ideas;

Terry Stewart for allowing me to debate the merits of Holographic Reduced Representations and lending his advice on various stages of my research;

Ian Mackinnon, Rick Theis, Nick Businger, Donna Craig, Andrew Dilts, Jeff Henry, David Herbert, Jonathan Naymark, Michael Garfinkle, Dov Zigler, John Anderson, Rob Warren, and Dave Hoff for helping keep my sanity, in a manner of speaking, throughout these past two years.

## Dedication

This is dedicated to my loving and supportive family, who can carry this thesis around with them as proof of a return on their investment in my education.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The amount of textual data in the world increases every day. In fact, with society's ubiquitous adoption of the Internet, the average person has an unprecedented level of access to text-based information. As people are exposed to increasing quantities of textual data, the risk of information overload and information paralysis becomes higher. As a result, we need to develop automated methods that structure the information to allow easy organization and retrieval of relevant documents.

This thesis studies representation techniques for *automatic text classification* [49]. The goal of text classification is to assign a text document to a predefined category based solely on the text contained within the document. Successful text classification is highly dependent on the representations used. A representation of a dataset that leaves out information regarding prominent features of the dataset will result in poor performance, no matter how good the classification algorithm may be. In the case of natural language text, there are many choices that must be made in converting the raw data to high-dimensional vectors for classification algorithms to process. The most traditional approach to text representation is to adopt the 'Bag-of-Words' (BoW) [42] document representation approach, where the semantic meaning of the document is taken to be the frequency counts of the words in the document. Alternatively, the 'Bag-of-Concepts' (BoC) document representation approach, first proposed by Sahlgren and Cöster [46], uses the intuition that the meaning of a document can be considered as the union of the meanings of the terms in that document. In this thesis, we present a new text representation approach, which assumes that a document's meaning is not only made up of the union of meanings of the individual words in that document, but also how those words are used grammatically within the document.

Because of the rapid rise in available of text-based information in the past few decades, much research attention has been focused on automatic text classification. Sebastiani [49] has compiled an excellent survey of automated text classification research, which we use to summarize the problem in the next section.

## 1.1 Text Classification

Before the advent of modern automated computational approaches to text classification, the task was traditionally carried out by humans. The ultimate goal of automated text classification is to mimic the process carried out by these human classifiers. Human classifiers are able to infer classification rules from a set of correctly classified example documents, and then use these inferred rules to classify unseen documents into the appropriate categories. This type of process is similar to the research undertaken by the machine learning research community, and it has been that community that has proposed a number of novel approaches for automated text classification. It is important to highlight the various different 'flavours' of text classification in the research, which we describe below.

The definition of *hard text classification* [49] is attempting to approximate the function $\check{\Phi}_h : \mathcal{D} \times \mathcal{C} \mapsto \{True, False\}$, where $\mathcal{D}$ is the set of documents, and $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ is the set of predefined classes. The set of given preclassified example documents from which to train the classifier is represented by $\Omega \subset \mathcal{D}$. The function $\check{\Phi}_h$ assigns a document $d_i$ to class $c_j$ iff $\check{\Phi}_h(d_i, c_j) = True$, otherwise $\check{\Phi}_h(d_i, c_j) = False$ and the document $d_i$ is not assigned to class $c_j$.

Similarly, *soft text classification* [49] is defined as attempting to approximate the function $\check{\Phi}_s : \mathcal{D} \times \mathcal{C} \mapsto [0, 1]$. The soft text classification function $\check{\Phi}_s(d_i, c_j)$ ascribes a confidence value to whether document $d_i$ is a member of class $c_j$. From the definition, it is easy to see that soft text classification can be used to simulate hard text classification by assigning a simple threshold value $\tau_i$ for each class $c_i$.

Text classification can also be divided into *multi-label* or *single-label* text classification. Multi-label text classification assigns a document to any $k \in \{0, 1, \ldots, |\mathcal{C}|\}$ number of classes. In multi-label text classification, as document can be classified as being a member of a number of different classes at the same time. Single-label text classification can be seen as a special case of multi-label text classification where $k = 1$, and the document is only assigned one class label. As well, *binary text classification* is a special case of single-label text classification, where a document $d_i$ is either assigned to class $c_i$ or its complement class $\overline{c_i}$. Sebastiani [49] notes that binary text classification is the most theoretically interesting case of text classification, since multi-label text classification can be created by combining $|\mathcal{C}|$ binary classifiers which classify under $\{c_i, \overline{c_i}\}$ to solve the larger classification problem of $\{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$. However, this approach assumes statistical independence amongst the classes, which may not be valid (e.g. hierarchical class structures). Other approaches (e.g. regression based approaches) have been proposed [49], which take into account possible dependence amongst classes. This thesis limits its investigations to hard, binary text classification.

Automatic text classification, whether hard or soft, is an attempt to use the approximate classifier $\Phi$ to estimate the true classification function $\check{\Phi}$ for a particular document domain $\mathcal{D}$. But the problem definition tells us nothing about how to represent a document in a manner suitable for classification, how to create the text classifier $\Phi$, or how to measure how effective the classifier $\Phi$ is in relation to the true classifier $\check{\Phi}$. More commonly, these topics are referred to as document representation [49], classifier learning and classifier evaluation. These topics can all be considered as individual phases in creating a text classification system. Although each phase has historically been studied in isolation from the others, it is not the case that each phase is independent from the others. For example, the method for document representation can have a large impact on the effectiveness of a classifier. This thesis focuses on representation methods since representations are fundamental for choosing appropriate classifiers.

## 1.2 Hypothesis, Motivation, and Delimitation

The main hypothesis of the thesis is that a text representation based upon linguistic structure can outperform representation techniques that ignore it. Although a significant amount of research in the literature has been devoted to text classification, most research has focused on the classification algorithms associated with these systems and less on the document representation techniques. Currently, most approaches to document representation for text classification adopt the Bag-of-Words (BoW) approach. BoW considers the frequencies of word occurrences as the most important features in a document and ignores the semantic relationship between words and the grammatical structure of the document. This is largely because past approaches that have tried to include more complex structures, or semantics, have been found lacking in their overall classification performance [26, 37].

However, we hypothesize that these negative conclusions are premature. For instance, consider the example of including semantic relationships between words in document representations. The work by Kehagias et al. [26] relies on explicit, hand encoded dictionaries to define semantic relationships between words, and does not improve BoW results. However, more recent work that employs automatically generated semantics using Latent Semantic Analysis and Random Indexing have been shown to be more effective than BoW approaches in some circumstances [46]. As a result, it seems more a matter of determining how best to include semantic relationships within a document representation, than of whether or not semantic relationships are useful for classification.

We assume that the same is true of including the syntactic structure of a document within a document representation. A recent comprehensive survey suggests that including a document's syntactic information will not help in classification [37]. However, the standard method for including a document's syntactic information is simply to add the syntactic information as a completely new, independent feature of the document. This new feature is then treated the same as each of the other identified features. In this thesis, we propose a representation technique that takes a very different approach to feature generation by distributing syntactic information across the document representation. The motivation behind this representation technique is that since the task of text classification is stated as classifying a text document based solely on the document's text, a representation that better models the linguistic structures of the text will improve classification scores.

Our primary motivation for our investigations is to compare the relative merits of various text representations. Although, there are other areas of text classification that are of great research interest (e.g. classifier learning), this thesis will not focus on them. As a result, we have not designed our experiments to be directly comparable with top classification scores seen in the literature. The different phases of a text classification system each have numerous parameters that can be tuned to produce higher final classification scores. Although it would be interesting to engage in an analysis of the highest assignable classification score that each representation we investigated can achieve, these experiments would have involved more parameter tuning and would have limited the scope of our research. As well, we have only investigated documents written in English.

## 1.3 Thesis Outline

The remaining chapters are organized to mirror the steps of a text classification system in the following way. Chapter 2 introduces different text representation approaches seen in the literature, and describes common preprocessing steps. Chapter 3 contains a survey of various dimensionality reduction techniques that are commonly applied to text representations. Holographic reduced representations and our new proposed text representation technique are described in Chapter 4. Chapter 5 introduces the Support Vector Machine classifier used in our classification experiments. Methods for evaluating classifiers are introduced and discussed in Chapter 6. Chapter 7 discusses the methods, the data, and the parameters used in the experiments. The experimental results and a discussion of them appears in Chapter 8. Finally, the main conclusions and some areas for future research are described in Chapter 9.

# Chapter 2

# Text Representations

In this chapter we introduce various base text representation and preprocessing methods designed for text classification. Our focus is on the vector space model for text representation, from which all of the representation techniques we describe are derived. A discussion of defining the feature space for text representations follows, where we introduce the specific representation techniques of Bag-of-Words and Bag-of-Concepts and discuss the strengths and limitations of these techniques. We then introduce a description of a common weighting scheme used to weight the dimensions of the feature space. Finally, we discuss some common text preprocessing techniques, which are used to prepare text for representation and classification.

## 2.1 The Vector Space Model

If we desire to classify natural language text documents, we must identify a process for transforming this textual data into a form that is suitable for classification. There are various mathematical foundations from which we can draw, specifically statistical probability, linear algebra, or set theory, to name a few. In this thesis, we adopt a linear algebraic mathematical basis for our representations. As a result, we must attempt to create a mathematical space in which to position our documents, which will allow us to exploit the research from the pattern recognition community in distance based classifiers. Ideally, similar documents in our document space will lie close to one another, and dissimilar documents will lie far away from each other in the space.

As first demonstrated by Salton et al. [48], we can set up our document space by defining each document $d_i \in \mathcal{D}$ as a vector $\underline{d}_i = \big(x_{i1}, x_{i2}, \ldots, x_{i|\mathcal{F}|}\big)$, where $\mathcal{F}$ is the feature set for the representation, and $x_{ij}$ is the contribution of feature $f_j$ to document $d_i$. Often, the frequency of word occurrence is used as the features for the representation. As well, the contribution of each feature to the representation of a specific document is usually calculated by applying a weighting scheme to the frequency counts of each feature. These weighting schemes will be described in further detail in Section 2.3.

We can represent the document set $\mathcal{D}$ as a $|\mathcal{D}| \times |\mathcal{F}|$ matrix, where each row in the matrix represents a document represented over the chosen $\mathcal{F}$-dimensional feature space. As we desired, we can analyze this space mathematically and construct various similarity measures, e.g. Euclidean distance or angles between document vectors, in order to measure the relative similarity of documents for classification.

## 2.2 Defining the Feature Space

Although the features of our representation can have any number of definitions, practically we desire that the features be defined so that they represent useful entities for classification and are easy to extract from the documents. In this case, 'useful' means that the feature set is able to accurately capture the content of the natural language text documents, and that each feature can be found and quantified in a document if present.

### 2.2.1 Bag-of-Words and Other Surface Form Representations

As suggested by Salton et al. [48], representation features are most often directly related to *surface forms*, the language elements readily available within documents without performing any deep processing. When we define our features as word stems, we create what is referred to as the 'Bag-of-Words' (BoW) representation. We can also use $n$-gram elements, specifically collections of $n$ successive word stems[1], to define our feature space. The motivation for using these larger elements is that they are assumed to be able to better capture the meaning of the documents. However, using these larger units as features have not shown any specific advantage in information retrieval tasks [49]. Joachims [17] defines features on the sub-word level, collections of successive characters in a word, in addition to what he calls the word level features (word stems), and multi-word level features ($n$-grams). The sub-word level can be useful for languages that contain many composite forms.

Using these three possibilities for defining our surface form features, we note that the word stem features represent a middle ground between $n$-gram features and sub-word features. Features defined at the sub-word level are more general, but introduce more ambiguity into the representations. Ambiguity, in turn, will introduce noise into the representations. However, certain machine learning techniques depend on the generality of this type of representation, i.e. a Naive Bayes classifier assumes that all features are statistically independent. Alternatively, features defined on the $n$-gram level are more comprehensive in their modeling of a document, but are more specific to particular documents and less applicable across a document corpus. This increase in specificity increases the amount of data required to accurately model the corpus statistically.

[1]$n$ is commonly chosen as $n = 2$ or $n = 3$

The idea that a document's semantic content can be modeled purely by surface form representations has been shown to be successful in a variety of applications [42]. For example, many prevalent information retrieval applications, such as search engines and document management systems, rely on these types of representations.

### 2.2.2 The Pitfalls of Linguistic Semantic Relationships

What makes modeling language such an interesting challenge for text classification is that natural language has not been formally modeled mathematically. Natural languages are not always clear, allowing a sentence to be parsed in different ways, leading to different meanings of the sentence. For example, the sentence "Terrorist head seeks arms." has two distinct meanings due to the fact that the words 'head' and 'arms' have multiple applicable meanings in the sentence. This type of ambiguity is a motivating example for word sense disambiguation in natural language processing research. In linguistics, this is defined as *polysemy*, where a word can have multiple different meanings.

Additionally, a single concept may have multiple words representing it in language. For example, the words 'cup' and 'mug' both represent the same semantic concept, a vessel that holds liquid and is designed for drinking. In linguistics, this type of relationship is called *synonymy*. Synonymy is not necessarily an equivalence relation and synonyms do not necessarily always represent the exact same semantic concept. However, it is important to recognize the similarity in the semantic concepts between synonyms to appropriately model natural language since it is highly prevalent in natural language use. Deerwester et al. [8] experimentally determined that two people will use different key words to describe a well-known object over 80% of the time.

Polysemy and synonymy relationships can be problematic when creating representations for text classification systems[2]. Surface form representations, such as the BoW representations, do poorly in dealing with synonymy and polysemy relations. Simple synonymy relations can be alleviated with word stemming, which identifies words coming from the same morphological root, explained further in Section 2.4. But more complicated synonymy representations, e.g. the 'cup' and 'mug' example provided earlier, will be ignored by surface form representations since, by definition, these representations avoid the linguistic processing needed to identify this kind of relation. The same is true for polysemy relations. Surface form representations may perform well insofar that synonymous and polysemous words occur in a higher frequency in a particular document class in comparison to other classes, but no real understanding of these linguistic concepts will be exploited in these representations.

[2]Polysemy tends to decrease precision, while synonymy tends to decrease recall in text classification systems. These concepts will be explained in Chapter 6.

### 2.2.3 Representing Word Meaning

We can also use a vector space to represent the semantic meanings of words. These types of vector spaces are typically referred to as *word-spaces*. We'll use the term *term context vector* to indicate vectors of this type. What follows is an overview of two techniques for creating term context vectors, based on the work of Lavelli et al. [31].

We consider two distinct methods for representing the semantic meanings of words in a vector space, namely the *document occurrence representation* (DOR) and the *term co-occurrence representation* (TCOR)[3]. Both of these methods are motivated by the distributional hypothesis of language, which states "words with similar distributional properties have similar meanings" [45]. We can rephrase this definition to state that words that appear in the same contexts will be semantically similar.

DOR representations represent a term $\underline{t}_i \in \mathcal{T}$ as $\underline{t}_i = \left(c_{i1}, c_{i2}, \ldots, c_{i|\mathcal{C}|}\right)$, where $c_{ij}$ represents the weighted frequency of term $t_i$ occurring in context $c_j \in \mathcal{C}$. From this definition of a term context vector, the meaning of a term can be considered simply as the bag-of-contexts in which the term occurs. If we define contexts to be the documents in which the term occurs, this representation is conceptually related to the BoW representation, e.g. just as BoW document representations are a collection of a term's frequency counts in that document, DOR term representations are a collection of a term's frequency counts across documents.

Alternatively, TCOR representations represent a term $\underline{t}_i \in \mathcal{T}$ as $\underline{t}_i = \left(t_{i1}, t_{i2}, \ldots, t_{i|\mathcal{T}|}\right)$, where $t_{ij}$ represents the weighted frequency of term $t_i$ occurring with another term $t_j \in \mathcal{T}$ in some context. This is usually accomplished by passing a set sized sliding window over a document, centred around a focus word, and counting word co-occurrences with the focus word. In this case, the meaning of a term can be considered simply as the bag-of-terms in which the word co-occurs in a defined context.

It is still an open question on how to define a 'context' in these representations. One method is to use the Generalized Vector Space model [55] or Latent Semantic Indexing (LSI) model [8, 29], both DOR models, which use the entire document as a single context and each term is a vector of the weighted counts in which it occurs in each document. Another method, the Hyperspace Analogue to Language model (HAL) [33], a TCOR model, uses individual words as contexts and each term is a vector of the weighted counts in which it co-occurs with other terms as determined by passing a fixed-size sliding window, ignoring sentence boundaries, over the document.

Choosing either DOR or TCOR has an overall effect on the size of the matrix these representations produce. DOR representation models will yield a matrix that is $|\mathcal{T}| \times |\mathcal{C}|$, which will vary in shape depending on the cardinality of $\mathcal{T}$ and

[3]DOR and TCOR representations are defined more formally in Section 3.3.3.

$\mathcal{C}$. Alternatively, TCOR representation models will always yield a square matrix $|\mathcal{T}| \times |\mathcal{T}|$.

Lavelli et al. [31] have shown that TCOR can outperform DOR in some word classification tasks. They argue that TCOR better identifies different words with identical meanings since it is likely that most authors will not tend to use terms with identical meaning identically throughout a document, but rather choose one of the words exclusively for a document. However, Sahlgren [43] argues that DOR or large window TCOR representations better capture associative relations between words, what the words are *about*, while small window TCOR representations better capture semantic relations between words, what the word *means.* This provides the intuition that DOR and large window TCOR representations are better for *topical information* tasks, such as text classification.

Recently, Karlgren et al. [24] have postulated that local structure in these word-space models are more important than their global structure. They suspect that word-spaces are structured less as homogeneous or lumpy subspaces, but as filamentary subspaces. As a result, the local structure of these word-spaces are where the useful semantic relations reside, such as the information about a word's meaning and information about its relation to other words. Although the conclusions in the context of text classification methods are currently unclear, these results seem to suggest that local analytic techniques of vector space structure may be more appropriate, as opposed to methods that examine the global structure.

### 2.2.4 Bag-of-Concepts

The Bag-of-Concepts (BoC) text representation is a more recent representation scheme, introduced by Sahlgren and Cöster [46] and meant to move beyond representing the mere surface forms of words. BoC representations, generated using either the DOR or the TCOR approach, are based on the intuition that the meaning of a document can be considered as the union of the meanings of the terms in that document. This is accomplished by generating term context vectors for each term within the document, and generating a document vector as the weighted sum of the term context vectors contained within that document. BoC has a better chance of preserving synonymy relations when compared to other surface form representations since this representation technique relies on term context vectors that can capture semantic relations between words.

BoC has been shown to have a classification advantage over surface form representations in certain situations [46]. Nevertheless, the BoC scheme still has its limitations. BoC ignores the large amount of syntactic data in the documents not captured implicitly through word context co-occurrences. For instance, although BoC representations can successfully model some synonymy relations, it can not model polysemy relations. For example, consider the word "can". Even though the verb form (i.e., "I *can* perform that action.") and the noun form (i.e., "The soup is in the *can.*") of the word occur in different linguistic contexts and have different meanings, the generated term vector for “can” will be a combination of these two meanings. As a result, the representation will not be able to model polysemy relations, especially polysemy relations involving a word that can be used in different parts of speech.

### 2.2.5 Computational Considerations

BoW representations tend to be quite sparse, since the dimensionality of the space is defined as the total vocabulary of a corpus and individual documents only use a small fraction of the total corpus vocabulary. We can construct these representations in a very computationally efficient manner since it is linear in the number of words in a document.

Constructing the BoC representations is slightly more computationally intensive, especially when large window TCOR representations are used. The computational complexity in constructing these representations is quadratic in the number of words in a document. We must also sum these term context vectors to create a final BoC representation, which will scale in runtime with the density of the vectors. Vector density will vary from one term context vector to another, but the average density is usually high. However, constructing DOR and small window TCOR representations present a better situation with respect to computational efficiency. In the DOR case, the representation is conceptually related to BoW representations; thus, the complexity will also be linear in the number of words in a document. Nevertheless, the term context vectors will still be dense with a corpus including sufficiently large and varied documents.

### 2.2.6 Moving Beyond Semantics: Parts-of-Speech

Limiting representations to word surface forms or word meanings based on context use or co-occurrence can only get us so far in creating an accurate model of natural language. Natural language is more than just the sum of words used; there are important syntactic and grammatical elements of language that give rise to its clarity and communicative power. Unfortunately, many natural language representations, such as BoW and BoC, are agnostic towards modelling these aspects of language. However, including this type of information can help model word polysemy relations that have been ignored by current representations.

*Part-of-speech* (PoS) information is one technique for including some of the inherent structure of language, without requiring a full understating of grammar. Manning and Schütze [34] present a summary of various approaches to determining a word’s PoS tag automatically based on the probability of a word’s use in a particular PoS and by recognizing common sentence structures. Early approaches to PoS tagging used Markov models and assumed that a word’s PoS tag can be determined by looking at the PoS tag of the word that immediately preceded it[4]. However, these methods require a large amount of hand-tagged corpora to train the models. Alternatively, hidden Markov model taggers were proposed to learn the distributions of tag sequencing without any tagged training data.

But the assumptions of Markov based techniques for PoS tagging, specifically that a word's tag can be determined by looking at a limited number of preceding tags, have limitations in PoS tagging. As a result, transformational-based learning approaches have been proposed for PoS tagging. These approaches condition the tagger on a wider range of lexical and syntactic regularities in language by tagging at first based on its most frequent use, and then using learned corrective transformations to achieve better tagging accuracy[4]. Although transformational taggers require training to learn the corrective transformations, the training is significantly faster than training Markov model taggers. As well, these taggers can be trained in a supervised or unsupervised fashion [5].

Manning and Schütze [34] state that most taggers in the literature report accuracy percentages between 95%-97%. Although these numbers are quite high, it is important to note that a 'dumb' tagger that tags all words as nouns can get an accuracy of 90% with stop words removed. As a result, high accuracy is crucial. However, another key consideration in selecting a tagger is its computational cost for both tagging and training.

Returning to our two example sentences, "I *can* perform that action," and "The soup is in the *can*,", we can use the different PoS tags of the word 'can' to differentiate the verb sense of the word used in the first sentence from the noun sense used in the second sentence. Processing the sentence in this way allows us to identify the polysemous senses of the word 'can'. However, we must consider how we can appropriately include this new information within a vector space representation model. Kehagias et al. [26] use word sense information to distinguish each word sense as independent surface forms. This approach yields similar classification results to not including this information. Moschitti and Basili [37] specifically use PoS information as independent features in their representations, but produce lower classification results than standard surface form representations. Clearly, a new method of incorporating PoS information in vector space representations is needed to achieve a classification benefit.

## 2.3 Feature Weighting

Certain words in a document will be more useful for differentiating documents for classification than others. As a result, it is standard practice to weight document

[4]Bigram (i.e. incorporating the tags of the *two* words that immediately precede our focus word) or trigram contexts for Markov model taggers have been created at the cost of added computational complexity [34].

terms based on their importance to a particular document. Considering the different representation techniques presented above, we must apply the term weighting differently depending on the representation used. In BoW representations, features are defined to be frequency counts of word-stems within a document; therefore, we apply our term weightings independently to each word frequency count. In BoC representations, term context vector features are defined to be a word's word or context co-occurrence frequency; therefore, we must apply our weighting to the entire term context vector.

Many weighting schemes have been proposed in text classification research [49]. These weighting schemes usually adopt the following three assumptions when determining a document word's weight. Firstly, a word that appears frequently in a document is most likely important to that document, denoted as the word's *term frequency*. Secondly, a term that appear rarely across all documents is most likely important to the document in which it appears, denoted as the word's *inverse document frequency*. Thirdly, the length of a document is not representative of the importance of that document. We can reduce the effects of document length through a normalization term for the document vector. These assumption can be summarized by equation 2.1, which represents the weight, $w_{ij}$, for word $t_j \in \mathcal{T}$ in document $d_i \in \mathcal{D}$ as,

$$w_{ij} = tf_{ij} \times idf_j \times norm_i, \tag{2.1}$$

where $tf_{ij}$ represents the term frequency, $idf_j$ represents the inverse document frequency, and $norm_i$ represents the normalization term.

We calculate the $tf_{ij}$ term using equation 2.2 below,

$$tf_{ij} = \#\left(d_i, t_j\right), \tag{2.2}$$

where the function $\#\left(d_i, t_j\right)$ returns the frequency count of word $t_j$ appearing in document $d_i$. The term $idf_j$ is calculated using equation 2.3 below,

$$idf_j = \log \frac{|\mathcal{D}|}{\#\mathcal{D}\left(t_j\right)}, \tag{2.3}$$

where the function $\#\mathcal{D}\left(t_j\right)$ returns the number of documents in which term $t_j$ appears. Finally, the term $norm_i$ is calculated using equation 2.4 below,

$$norm_i = \frac{1}{\sqrt{\frac{1}{\#\mathcal{T}(d_i)} \sum_{k=1}^{|\mathcal{T}|} \left(tf_{ik} \cdot idf_k\right)^2}}, \tag{2.4}$$

where the function $\#\mathcal{T}\left(d_i\right)$ returns the number of words contained in document $d_i$. Conceptually, the $norm_i$ term is normalizing the document vectors based on document length and then normalizing the vector to unit length.

Modifications to this common weighting scheme have been proposed in the literature. Sebastiani and Debole [50] have proposed calculating inverse document frequency based only the specific category for which we are designing a classifier.

As well, Yang and Pederson [58] note that there is a high correlation between a document's category assignment and the document frequency of a word contained in that document. However, our research does not focus primarily on text weighting schemes, and as such, we will adopt the simpler and more common weighting scheme previously described.

## 2.4 Preprocessing

Before we build representations of our documents for a particular text corpus, we first perform some common preprocessing tasks on the text data. Some preprocessing techniques remove *stop words* (e.g. 'the', 'a', 'an', etc.) from a document or remove words that appear with very low frequencies. We have chosen to avoid these types of preprocessing steps as we find that they are not well-motivated and have been proposed as ad-hoc methods to ease computational constraints. In our research, we focus our preprocessing steps on *morphological preprocessing*, that is removing a word's inflectional affixes. Removing inflectional affixes has the effect of helping to identify relationships between words, since many words derived from the same morphological root will have similar meanings (e.g. attach and attachment are both derived from the morphological root attach and have related meanings). Morphological preprocessing can help determine more general features that are relevant to a large proportion of documents for our representation techniques. There are two main approaches to morphological preprocessing, namely *stemming* and *lemmatization*.

Stemming is the simpler of the two methods that incorporates simple transformations to strip affixes. Since this method does not take into account any of the linguistic use or lexical information available, the method does not lead to high accuracy in identifying the correct morphological root of a word. As a result, although stemming may be able to help in identifying word relations, it can also introduce polysemy in our final representations. This can be overcome by customizing the stemmer to a particular document domain. Nevertheless, since stemming relies on basic transformations, the process is not very computationally intensive or hard to implement.

Lemmatization is a more complicated approach based in more detailed linguistic processing of a word to determine its morphological root. Usually, lemmatizers are implemented using a combination of transformational rules and a lexicon, which sometimes contains handwritten mappings of a word to its inflectional root. The disadvantage to this method is that there can be many inflectional forms for words that need to be identified and listed, particularly in languages other than English.

We have taken a stemming approach to morphological preprocessing in this thesis. We will use the Porter Stemmer, a popular stemmer in natural language processing research, to preprocess all of our data. In practice, stemming has proven sufficient for identifying morphological roots in English. In contrast, lemmatization may be more appropriate when applied to heavily inflected languages.

# Chapter 3

# Dimensionality Reduction

Once text data has been preprocessed and converted to a base representation, the dimensionality of the representations must be reduced to ensure the data is suitable for our classification methods. This chapter discusses dimensionality with respect to text representations and various techniques to manage and reduce the dimensionality of these representations. We have split this section into three sections. The first section describes the problems with dealing with representations of high dimensionality and motivates the need for dimensionality reduction techniques. The feature selection approach for dimensionality reduction is described in the second section. Finally, the third section describes feature extraction approaches to dimensionality reduction with a focus on the Random Indexing technique.

## 3.1 The Curse of Dimensionality

The text representations that we introduced previously in Chapter 2 typically create a vector representation space of extremely high dimensionality. The preprocessing techniques previously discussed in Section 2.4 can reduce the dimensionality of the representations somewhat. However, the resulting feature space is still typically on the order of 100 000 dimensions, depending on the document collection. Representations with dimensionality of this magnitude pose a significant computational barrier to many machine learning classification algorithms, rendering the classification problem computationally intractable to some algorithms. Many classification algorithms have a computational performance that scales very poorly with dimensionality.

Textual data has inherent difficulties in terms of accurate feature sampling. As explained in Section 2.2.2, synonymy and polysemy relations within the text hinder accurate feature sampling within a text document and lead to sampling noise. As well, the large number of low frequency occurring terms within a text document further aggravates this sampling noise by possibly introducing sampling errors within the representations. This sampling noise can play havoc with many machine learning classification algorithms by increasing the likelihood that the algorithm will *overfit* the data. Overfitting occurs when a classification algorithm creates an overly complicated model of the training data by incorporating noise within the data as relevant model features leading to loss of generality and poor classification performance. Sebastiani [49] argues that the number of dimensions in the data should be proportional to the number of training documents to avoid overfitting.

We can reduce the effects of these problems with dimensionality by reducing the dimensionality of the data. This reduction can either be accomplished by *feature selection* methods, where a certain number of data dimensions are selected as the most important for the data, or by *feature extraction* methods, where a smaller number of new features are created for the data by combining information in the original dimensions of the data.

## 3.2 Feature Selection

The goal of feature selection is to keep a subset $\mathcal{F}' \subset \mathcal{F}$ of the original features of the data and use this feature subset for classification. Feature selection has the advantages that the computational requirements for the selection of the features is straightforward and the original units of each feature are preserved in the feature subset. The problem with feature selection is that we are attempting to solve a difficult optimization over the discrete space of original features, unlike feature extraction which attempts to solve a similar but easier optimization problem over a continuous space, which we will see in Section 3.3. There are two main approaches that can be taken to feature selection; we can adopt either a *filtering* approach or a *wrapper* approach.

### 3.2.1 Filtering

In the filtering approach to feature selection, we attempt to select a subset of features of the data for classification based on some objective criterion. This criterion is independent of the classification algorithm used. In selecting a subset of features we want to ensure that our subset is appropriate; that is to say, we want to make sure that our feature subset contains the maximum amount of information about our data, considering its limited size. Therefore, in the literature it is typical to see information theory metrics used, such as *Information Gain* (IG) [58] and *Gain Ratio* (GR) [50].

We can define the amount of information contained within an event $e_i$ as the negative logarithm of its probability,

$$I(e_i) = -\log \Pr(e_i). \tag{3.1}$$

Therefore, if $e_i$ happens frequently, there is little information associated with the event. However, if $e_i$ seldom occurs, then a lot of information is associated with the event when it occurs. From this understanding, term entropy for a random variable $e$ can be defined as

$$H(e) = \sum_i -\Pr(e_i) \log \Pr(e_i). \tag{3.2}$$

We can treat entropy as an expression of the degree of randomness of a random variable $e$, thus entropy is maximized when all events are equally likely and is minimized when only one event is guaranteed. Using these definitions we can now investigate the amount of information conveyed by one event about another event. In our case these events are the information of a particular data feature $f_i \in \mathcal{F}$ about membership in a class $c_i \in \mathcal{C}$. We call this measurement *mutual information*, and we define it as

$$I(c_i; f_i) = \log \frac{\Pr(c_i, f_i)}{\Pr(c_i)\Pr(f_i)}. \tag{3.3}$$

From this definition we can see that if feature $f_i$ is independent of class $c_i$, then the mutual information is zero. However, if the feature $f_i$ is sufficient on its own to predict membership of class $c_i$, then the mutual information between the two is just the information of the class, $I(c_i)$.

Now we can define Information Gain and Gain Ratio using these definitions. Information Gain for binary classification can be considered the average amount of mutual information of our features and our classification variables. This can be considered the difference of the entropy for our binary class variable and the entropy of our class variable given our feature variable. This value is defined as

$$IG(c_i; f_j) = \sum_{x \in \{c_i, \overline{c_i}\}} \sum_{y \in \{f_j, \overline{f_j}\}} \Pr(x, y) \log \frac{\Pr(x, y)}{\Pr(x)\Pr(y)}, \tag{3.4}$$

where $\overline{c_i}$ denotes non-membership in a class and $\overline{f_j}$ denotes that feature $f_j$ is not present in the data. We will define Gain Ratio as information gain normalized over the entropy of our binary classification variable:

$$GR(c_i; f_j) = \frac{IG(c_i; f_j)}{H(y)}, \tag{3.5}$$

where $y \in \{f_j, \overline{f_j}\}$ once again.

Information Gain and Gain Ratio can be used to rank the data features for their suitability for use in a feature subset. This ranking can be calculated either as the weighted average of the metric over all classes, or the maximum of the metric applied to a particular class. Therefore, if we want to select the $n$-best features we could take the $n$ features with the highest rank according to our chosen criterion. Unfortunately, the best ranking $n$ features are not always the most effective set of $n$ features. To illustrate this point, two features may individually contain a lot of information for a particular class, but using both features in a feature subset is not guaranteed to double to amount of useful information, since some of the information between the two features may be the same.

Returning to text classification, Yang and Pederson [58] show that in BoW text representations the document frequency of a particular term is highly correlated with the average information gain for the feature associated with that term over all classes. Sebastiani and Debole [50] show that it is most effective to take the maximum of either information gain or gain ratio of a feature applied to a particular class as the ranking for the feature.

### 3.2.2 Wrapping

Filtering approaches to feature selection measure the usefulness of individual features in a feature subset, but are unable to evaluate the usefulness of an entire subset of features. Alternatively, wrapper approaches to feature selection attempt to create the maximally useful subset of features for classification, regardless of the independent usefulness of a single feature. Kohavi and John [28] define the optimal feature subset as the subset of features $\mathcal{F}' \subset \mathcal{F}$, such that the performance of a given classification algorithm is optimal with respect to a given classification evaluation measure.

Wrapper approaches to feature selection must take the classification algorithm into account when selecting the optimal feature subset. This is accomplished by using the classification algorithm directly to evaluate all possible $\binom{|\mathcal{F}|}{n}$ feature subsets and choosing the subset that has the maximal performance given a classification evaluation measure. Unfortunately, this search can be prohibitive if $|\mathcal{F}|$ and $n$ are large since there will be $2^{|\mathcal{F}|}$ possibilities. Different search techniques, such as hill climbing algorithms and genetic algorithms, have been used to explore this search space, but their effectiveness tends to match filtering approaches for feature selection with a much higher computational cost.

## 3.3 Feature Extraction

Feature extraction takes a different approach to reducing the dimensionality of our data than feature selection. Rather than select a subset of features from the original data and ignore the rest as in feature selection, feature extraction attempts to transform the original feature space into a new feature space of reduced dimensionality using a transformation function that effectively separates the data. The new reduced feature space should represent the original information contained in the original space, while attempting to limit the noise introduced from the transformation. A notable consequence of feature extraction techniques is that the new features in the reduced feature space will not be directly interpretable, unlike the original features which corresponded to some measurement of the data.

### 3.3.1 Latent Semantic Indexing

One of the most common feature extraction dimensionality reduction techniques applied to text classification is *Latent Semantic Indexing*[1] (LSI) [8]. LSI uses a *singular value decomposition* (SVD) of the original feature space to transform that space into a new feature space of lower dimensionality. This is accomplished by selecting $k$ singular values and their corresponding singular vectors in order to obtain a rank-$k$ approximation of the original feature space [34].

When specifically applied to text representations for classification, LSI reduces the dimensionality of a term frequency-inverse document frequency weighted term-by-document matrix for the classifier training data, and then uses the resulting decomposition to transform the testing examples into the new feature space. Using this rank lowering method, Sebastiani [49] points out that multiple features from a particular class that indicate class membership can all be mapped to a single feature in the new space. Unfortunately, this comes at the cost that a single feature from a particular class that greatly indicates class membership will be weakened in the new space. Also, by using this rank lowering method on text data, LSI can alleviate some of the problems around synonymy relations in language. It is expected that these merged terms from the original space will have similar meanings.

The main problem with LSI is the computational cost associated with SVD. The computational complexity of SVD on an $m \times n$ matrix is $O(mnc)$, where $c$ is the average number of non-zero entries per row. When applied to text data as in LSI, this computational cost becomes $O(|\mathcal{F}||\mathcal{D}|c)$, where $c$ now represents the average number of features per document.

### 3.3.2 Random Mapping

*Random Mapping*[2] (RM) is a more computationally efficient method of reducing dimensionality. This technique maps the original high-dimensional feature space to a lower dimensional space using a random matrix with unit length columns. For example, if we were to multiply a large $F_{m\times n}$ matrix by a random $R_{n\times k}$ matrix, with $k \ll n$, we would end up with a resulting matrix $T_{m\times k}$:

$$T_{m\times k} = F_{m\times n} R_{n\times k}. \tag{3.6}$$

Bingham and Manilla [3] have shown that RM is both computationally efficient and sufficiently accurate for dimensionality reduction when applied to text and image data.

The mathematical motivation behind RM is the Johnson-Lindenstrauss lemma [20]. The lemma states that if we project points into a random subspace of sufficiently high dimensionality, we will approximately preserve the distances between

[1] Also commonly referred to as Latent Semantic Analysis.

[2] Also known as Random Projection.

the points. Frankl and Maehara [14] have developed a simple proof of the lemma. However, equation 3.6 is not strictly a projection since $R_{n\times k}$ is not guaranteed to be orthogonal. In fact, when $R_{n\times k}$ is not orthogonal the data can be significantly distorted. Rather than performing the computationally costly step of orthogonalizing $R_{n\times k}$, we can use a result presented by Hecht-Nielsen [15]. Hecht-Nielsen shows that there are more nearly orthogonal directions than true orthogonal directions in a high dimensional space, meaning that vectors that have random directions might be sufficiently close to orthogonal. More formally, this result suggests that $R^\top R$ will be approximately equal to the identity matrix. Bingham and Manilla [3] show that the mean squared difference between $R^\top R$ and the identify matrix is on the order of $1/k$ per element. Since $k$ tends to be large in applications of this method, the deviation from the identity matrix is insignificant and $R_{n\times k}$ is sufficiently orthogonal.

We can now turn our attention to how to construct this random matrix used for our projection. The simplest case would be to construct the random matrix along a normal distribution, with a mean of 0 and variance of 1. Achlioptas [1] notes that this real valued and dense matrix would not provide for the most optimal computational scenario when performing the projection. He presents an alternative method of constructing the matrix, where the elements $r_{ij}$ of the random matrix $R_{n\times k}$ are defined as

$$r_{ij} = \sqrt{\frac{3}{k}} \times \begin{cases} +1 & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -1 & \text{with probability } \frac{1}{6} \end{cases} . \tag{3.7}$$

Practically, the scaling factors are not incorporated until the projection has been performed; therefore, $R_{n\times k}$ only has elements $r_{ij} \in \{+1, 0, -1\}$. This matrix construction scheme leads to computational savings over a normally distributed random matrix since we can rely exclusively on integer arithmetic to perform the projection, as well as allowing us to process only the third of the elements that are non-zero while ignoring the rest. Sahlgren and Karlgren [47] generalize this construction method for a given dimensionality $k$ and a given density factor $\epsilon$ as

$$r_{ij} = \sqrt{\frac{1}{\epsilon}} \times \begin{cases} +1 & \text{with probability } \frac{\epsilon/2}{k} \\ 0 & \text{with probability } \frac{k-\epsilon}{k} \\ -1 & \text{with probability } \frac{\epsilon/2}{k} \end{cases} . \tag{3.8}$$

In practice any zero mean, unit variance distributions for $r_{ij}$ would yield a mapping that satisfies the Johnson-Lindenstrauss lemma. As a result, the density factor $\epsilon$ must be an even integer to ensure equal distribution of +1's and -1's.

Computationally, random mapping is very simple to perform. The random matrix $R_{n\times k}$ is easy to create and the projection can be accomplished on the order of $O(nkm)$. However, when the matrices are sparse, the projection can be computed on the order of $O(ncm)$, where $c$ is the number of non-zero entries per row. Applying random mapping to our text classification problem, this computational cost becomes $O(|\mathcal{F}'|c|\mathcal{D}|)$, where $c$ now represents the average number of features per document and $\mathcal{F}'$ is the reduced feature set. Compared to the $O(|\mathcal{F}|c|\mathcal{D}|)$ computational complexity of LSI, random mapping can be considered more efficient since $|\mathcal{F}'| \ll |\mathcal{F}|$.

### 3.3.3 Random Indexing

Following Kanerva's work on sparse distributed memory [22], Karlgren and Sahlgren [25] and Sahlgren and Cöster [46] have proposed an incremental method of performing this random projection for text data. RM would first build a $m \times n$ matrix of term document occurrences (DOR) or term co-occurrences (TCOR), and then project this matrix into a lower dimensional space using the random $R_{n \times k}$ matrix. However, performing this large matrix multiplication can be costly in terms of memory requirements, since we must store these large matrices in memory to perform the projection.

The *Random Indexing* (RI) technique [44], in contrast, assembles this lower dimensional matrix incrementally. In RI, we first create $k$-dimensional random labels for each dimension in our data[3], where $k$ is significantly less than the total number of dimensions in the data. For DOR representations, each time a term occurs in a document, that document's random label is added to the context vector for the term. For TCOR representations, each time a term co-occurs with another term, its random label is added to the other's term context vector. These sparse random labels are created by randomly distributing a small number of +1s and -1s amongst the elements of the vector and setting the rest of the elements to 0. Algorithm 3.1 and algorithm 3.2 outline the indexing process for DOR and TCOR contexts, respectively.

**Algorithm 3.1** Random Indexing - DOR

```
for all documents $d \in \mathcal{D}$ do
  $\underline{l}_d \leftarrow$ CreateLabel($k, \epsilon$) {Create an $\epsilon$-dense $k$-dimensional random label}
  for all term occurrences $t \in d$ of term $t \in \mathcal{T}$ do
    $T_t \leftarrow T_t + \underline{l}_d$ {Add random label to row $t$ of matrix $T$}
  end for
end for
```

The advantage of RI is that it is an incremental approach, meaning that context vectors can start to be created without sampling all the data. This is accomplished by distributing the step of the matrix multiplication required in RM. For example, if we only generate the context vectors for the first $t$ terms in a corpus using RI, the result would be equivalent to RM if we would have stopped building the large $m \times n$ matrix at the $t$-th term and then projected it into the lower random space.

[3]For DOR representations this would correspond to a document, while in TCOR this would correspond to a unique term.

**Algorithm 3.2** Random Indexing - TCOR

```
for all documents $d \in \mathcal{D}$ do
    for all term occurrences $t \in d$ of term $t \in \mathcal{T}$ do
        for all term occurrences $t' \in n$-sized context window centred on $t$ do
            if $\underline{l}_{t'}$ does not exist then
                $\underline{l}_{t'} \leftarrow$ CreateLabel($k, \epsilon$) {Create an $\epsilon$-dense $k$-dimensional random label}
            end if
            $T_t \leftarrow T_t +$ WeightingScheme($\underline{l}_{t'}$) {Add weighted random label to row $t$ of matrix $T$}
        end for
    end for
end for
```

Both RM and RI avoid the computationally costly singular value decomposition as utilized in LSI, while still significantly reducing the dimensionality of the data space. But more importantly, RI avoids constructing the large context count matrix required in RM. In RI, we only have to store the $n$, $\epsilon$ dense, $k$-dimensional random labels with the $m$, approximately $k$ dense, $k$-dimensional context vectors. As a result, using RI in place of other dimensionality reduction methods in constructing the term context vectors provides significant memory and processing savings.

# Chapter 4

# Holographic Reduced Representations

After dimensionality reduction is performed on our base text representations, more complicated representations can be built to better reflect the linguistic structure of the text. This chapter introduces the concept of Holographic Reduced Representations (HRRs) and illustrates how we are able to to use HRRs to augment traditional text vector space models with structural information. The work in this chapter is largely based on Plate's original work on HRRs [39]. The first section of this chapter introduces the concept of reduced distributed representations and notable past representation approaches. The next section focuses on Holographic Reduced Representation with an overview of the mathematical operations associated with the representations and an analysis of the vector space the representations create. The chapter ends with a section outlining the process of how we use HRRs to form our novel text representation.

## 4.1 Distributed Representations

Much of the work in distributed representation approaches has come from research into *connectionist models* for advanced cognitive or artificial intelligence. Connectionist models are typically made up of many simple processing units connected to each other in a network fashion, as illustrated in Figure 4.1. Data flows into these simple units from afferent connections from other units, simple processing is performed on this combined data, and the processed data is sent to upstream units through the unit's efferent connections. While early connectionist models showed a great deal of success for simple tasks, it became clear to the research community that attention needed to be paid to how to represent more complex forms of data for higher-level processing [39]. Representations play a key role in connectionist models and determine what a model can or cannot compute.

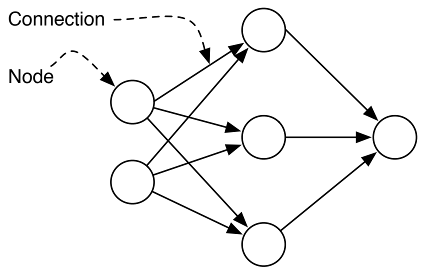


Figure 4.1: A Simple Connectionist Model.

### 4.1.1 Definitions

There are two main types of representations for connectionist models: *localist representations* and *distributed representations.* Localist representations use specific units to represent particular concepts in the model. For example, an object identification connectionist model would have a unit dedicated to the concept of 'chair', another unit dedicated to the concept of 'table', etc. Localist models have the advantage of having a simple method to explicitly represent concepts with an interface that is easy to use for the experimenter, but the technique explodes when large sets of objects or complex structures need to be represented. Alternatively, a distributed representation would represent the concept 'chair' in our connectionist model across a fixed number of distinct units. Distributed representations have the advantages of being analogical (i.e. similar objects will result in similar representations), continuous, and derivable from the statistics of relevant data, but distributed representations have difficulty representing hierarchical structure and make it hard to identify why a particular object has a specific representational pattern. The selection of a representation technique is highly dependent on the purpose of the representation, with localist representation used more heavily in language-related models while distributed representation are used more frequently in perceptual models.

A final important representation concept is the idea of *reduced representations.* Hinton [16] describes reduced representations as a method for encoding complex structures in distributed representations. Rather than explicitly store an entire hierarchical structure in a single representation, we can collapse, or reduce, branches of our hierarchy into a single level. We can then expand the reduced representation and reconstruct the branches as needed. Reduced representations are therefore compressed versions of the original data. This method allows us to store hierarchical data in a hierarchical way using a flat, fixed-sized, distributed representation.

Armed with these definitions, the following sections introduce past approaches for distributed representations that have been used to represent high-level predicate hierarchies like language.

### 4.1.2 Tensor Products

First described by Smolensky [51], *tensor-product algebra* can be used as a framework for representing hierarchical structures in a distributed representation. Specifically, tensor products provide a mechanism for binding vectors together that can be considered as a generalized outer product. For example, given two $n$-dimensional vectors $\underline{x}$ and $\underline{y}$, the tensor product $T = \underline{x} \otimes \underline{y}$ is equal to the outer product $\underline{x}\underline{y}^\top$. $T$ contains $n^2$ elements, with $T_{ij} = x_i y_j$, and is therefore considered a second-order tensor product. Higher and lower order tensor products also exist: a zero'th-order tensor product is a scalar, a first-order tensor product is a vector and a third-order tensor product is the product of three vectors with $n^3$ resulting elements. Similarly, a $k$'th-order tensor product is the product of $k$ $n$-dimensional vectors resulting in $n^k$ elements. Tensor products can be iteratively constructed, for example a fifth-order tensor product can be constructed as the product of a third-order and a second-order tensor. Tensor products can be decoded by using the inner product with the constituent vectors. For example, given the third-order tensor $T = \underline{x} \otimes \underline{y} \otimes \underline{z}$, under appropriate normalization conditions on the vectors, the vector $\underline{z}$ can be decoded from $T$ by $T \cdot (\underline{x} \otimes \underline{y}) = \underline{z}$.

Smolensky [51] also shows how tensor products can be used to model role-filler representations for predicate hierarchy representations. For example, a predicate $p(\underline{a}, \underline{b})$ can be represented as the sum of two role-filler products: $\underline{r_1} \otimes \underline{a} + \underline{r_2} \otimes \underline{b}$. Smolensky also proposes two methods for decoding these vectors: one that requires the roles to be linearly independent and provides an exact decoding, and another that allows for linearly dependant vectors and provides an approximate decoding that must be cleaned up with a post-processing system. Dolan and Smolensky [9] proposed an alternate method of representing predicates as third-order tensor products. Specifically given the predicate $p(\underline{a}, \underline{b})$, the resulting tensor product representation would be $\underline{p} \otimes \underline{r_1} \otimes \underline{a} + \underline{p} \otimes \underline{r_2} \otimes \underline{b}$, where $\underline{p}$ is a vector marker for the predicate. This method is actually equivalent to the tensor product of the predicate name with the previous method, since $\underline{p} \otimes \underline{r_1} \otimes \underline{a} + \underline{p} \otimes \underline{r_2} \otimes \underline{b} = \underline{p} \otimes (\underline{r_1} \otimes \underline{a} + \underline{r_2} \otimes \underline{b})$.

Hierarchical structures can be represented with tensor products when higher-order tensor products are used as fillers. For example using simplistic linguistic processing, we can define a noun phrase (NP) as a determiner and a noun, a verb phrase (VP) as a verb and a noun, and a sentence (S) as a noun phrase and a verb phrase. Using the example sentence "The boy went home." and assuming we have vectors created for our sentence words and phrase roles, we can formally represent these phrases in tensor products as $NP = \underline{r_{DET}} \otimes \underline{the} + \underline{r_{NOUN}} \otimes \underline{boy}$ and $VP = \underline{r_{VERB}} \otimes \underline{went} + \underline{r_{NOUN}} \otimes \underline{home}$. Moving up in the hierarchy, we can now represent a sentence as a tensor product as $S = r_{NP} \otimes NP + r_{VP} \otimes VP$. This example elegantly highlights the dimensionality explosion weakness of tensor product representations. In this example, our phrase representations are both second-order tensor products, but our sentence representation is a third-order tensor product. As the depth of the structure encoded in a tensor product grows, the dimensionality grows exponentially to represent the structure.

### 4.1.3 Associative Projective Neural Networks

Rachkovskij and Kussul [40] proposed *Associative Projective Neural Networks* (APNNs), a reduced representation, to exploit the finding that memory schemes are more efficient when stored as sparse binary vectors [22]. Although sparse distributed representations have advantages over denser representations, they are not easy to use since their composition operations must be performed in a manner that preserves the sparseness of the representation. APNNs overcome this problem by using carefully constructed operations that preserve the representation's sparseness, rather than binding and superposition operations.

In APNNs, each composition operation, used to bind multiple vectors together, is followed by a context dependent thinning operation, which brings the density of the resulting vector back down to the density of its inputs. Different context dependent thinning operations are chosen depending on the level of nesting in the representation. Specifically, given a dense vector $\underline{x}$ and a target sparseness of $n$ ones, a context dependent thinning operation removes some number of ones from $\underline{x}$ to gain a sparseness of $n$ in a manner that depends only on $\underline{x}$. Many context dependent thinning operations can theoretically exist, but Rachkovskij and Kussul [40] describe a process called additive thinning. Additive thinning uses a series of permutations to thin a vector $\underline{x}$ by ${}^{i}\langle\underline{x}\rangle = \underline{x} \wedge ({}^{i}P_1\underline{x} \vee {}^{i}P_2\underline{x} \vee \ldots \vee {}^{i}P_k\underline{x})$, where ${}^{i}P_j$ are matrices that permute elements of a vector, ${}^{i}\langle\cdot\rangle$ is the context dependent thinning operator at nesting level $i$, and $k$ is chosen dependent on $\underline{x}$ so that the density of ${}^{i}\langle\underline{x}\rangle$ is $n$. The matrices ${}^{i}P_j$ are arbitrary random permutation matrices, but must always be the same for the same $i$ and $j$.

We can use APNNs to represent high-level predicate and linguistic models using the same role-filler relationships that we saw previously with Tensor Products. For example, we will use another simplistic linguistic model to encode the sentence "Jon eats bacon." Using a simple agent-object linguistic model and assuming that we have previously generated vectors for our sentence words and roles, we can encode this sentence using APNNs as $S = {}^{2}\langle\underline{eats} \vee {}^{1}\langle\underline{eats_{agt}} \vee \underline{Jon}\rangle \vee {}^{1}\langle\underline{eats_{obj}} \vee \underline{bacon}\rangle\rangle$, where ${}^{1}\langle\cdot\rangle$ and ${}^{2}\langle\cdot\rangle$ are the nesting level context dependent thinning operators. Rachkovskij and Kussul [40] have shown that APNNs have similar representation power for hierarchy and structure as HRRs (explained in section 4.2) and Binary Spatter Codes (explained next).

### 4.1.4 Binary Spatter Codes

Kanerva [23] proposed *Binary Spatter Codes* as a method to encode complex composition role-filler structures using reduced binary distributed representations. Binary Spatter Codes are dense binary vectors that use element-wise exclusive-or as a binding operation and a thresholded sum as a superposition operation. Under these binding and superposition operations, the resulting dimensionality of the Binary Spatter Codes does not increase as long as the two input vectors are of the same dimensionality. Binary Spatter Codes can be used to encode hierarchical structures by using bound role-filler structures, or a superposition of multiple role-filler structures as a filler to a higher level role, as we saw in Tensor Products. But unlike Tensor Products, where encoding hierarchy exploded the dimensionality of the representation, Binary Spatter Codes keep dimensionality constant since its binding and superposition operations also preserve the dimensions of the inputs.

Plate [39] shows that Binary Spatter Codes have similar encoding properties to HRRs. Specifically, Plate shows that Binary Spatter Codes and APNNs are equivalent to HRRs in the frequency domain (using a Fast Fourier Transform) with constant power in each frequency and phase angles quantized between 0 and $\pi$.

## 4.2 Holographic Reduced Representations

Plate [38] introduced *Holographic Reduced Representations* to overcome many of the deficiencies of previously proposed distributed representations in the literature. HRRs are an implementation of Hinton's [16] reduced representations and can encode complex hierarchical structures using role-filler representations for complex structures. What separates HRRs from previous distributed representations is that they are based on mathematical convolution operations for composition that does not increase the dimensionality of the vector space. The main advantages for using HRRs is that they are simple to construct and easy to analyse. Like Tensor Products, HRRs are constructed using simple operations, making the properties of the representations easy to understand. HRRs also allow for simple and efficient representation of structure, freeing up computational resources for other resource intensive tasks. HRRs are more desirable to use over APNNs and BSCs due to their computational simplicity and ease to implement.

### 4.2.1 Circular Convolution

Ordinary convolution is a recursively applicable associative operator that can be used to successfully represent hierarchical structures in a distributed representation. However, ordinary convolution tends to increase the dimensionality of the input vectors making it an impractical method for representing deep hierarchical structures. *Circular convolution* [39], on the other hand, is a convolution operator that does not increase vector dimensionality, making it an excellent candidate for representing hierarchical structures. Like ordinary convolution, circular convolution can be considered the compression of the outer product of two vectors.

Circular convolution can simply be thought of as a multiplication operator for vectors. Circular convolution binds two vectors $\underline{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\underline{b} = (b_0, b_1, \ldots, b_{n-1})$ to give $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ where $\underline{c} = \underline{a} \circledast \underline{b}$ with

$$c_j = \sum_{k=0}^{n-1} a_{k \bmod n} b_{(j-k) \bmod n} \tag{4.1}$$

for $j = 0, 1, \ldots, n-1$. Like scalar and matrix multiplication, circular convolution is commutative: $\underline{x} \circledast \underline{y} = \underline{y} \circledast \underline{x}$; associative: $\underline{x} \circledast (\underline{y} \circledast \underline{z}) = (\underline{x} \circledast \underline{y}) \circledast \underline{z}$; and bilinear: $\underline{x} \circledast (\alpha \underline{y} + \beta \underline{z}) = \alpha \underline{x} \circledast \underline{y} + \beta \underline{y} \circledast \underline{z}$. Circular convolution also has an identity vector: $\underline{I} \circledast \underline{x} = \underline{x}$, $\underline{I} = (1, 0, 0, \ldots)$; and a zero vector: $\underline{0} \circledast \underline{x} = \underline{0}$, $\underline{0} = (0, 0, 0, \ldots)$. For almost all vectors, an exact inverse exists yielding $\underline{x}^{-1} \circledast \underline{x} = \underline{I}$. In algebraic expressions, circular convolution is given the same precedence as multiplication (i.e. $\underline{x} \circledast \underline{y} + \underline{z} = (\underline{x} \circledast \underline{y}) + \underline{z}$) and higher precedence than the dot product (i.e. $\underline{x} \circledast \underline{y} \cdot \underline{w} \circledast \underline{z} = (\underline{x} \circledast \underline{y}) \cdot (\underline{w} \circledast \underline{z})$).

Circular convolution also has an approximate inverse, *circular correlation*, which can also be regarded as a compressed outer product. Given two vectors $\underline{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\underline{b} = (b_0, b_1, \ldots, b_{n-1})$, we can use circular correlation to give $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ where $\underline{c} = \underline{a} \oslash \underline{b}$ with

$$c_j = \sum_{k=0}^{n-1} a_{k \bmod n} b_{(j+k) \bmod n} \tag{4.2}$$

for $j = 0, 1, \ldots, n-1$. If we have a *trace vector*, that is a convolution of some cue vector with another vector, $\underline{t} = \underline{c} \circledast \underline{v}$, then circular correlation on $\underline{c}$ with $\underline{t}$ will yield a vector that is a distorted version of $\underline{v}$, $\underline{y} = \underline{t} \oslash \underline{c}$ with $\underline{y} \approx \underline{v}$. A condition for circular correlation to approximately decode circular convolution is that the elements of each $n$-length vector must be independently and identically normally distributed with a mean of 1 and a variance of $1/n$.

It is not intuitive that a convolution trace of length $n$ can store several $n$-length vectors inside of it. This can in fact occur since circular convolution stores these vectors in a trace with lower fidelity. Only the information required to recognize a vector within a trace is stored within that trace, but there is not enough information to accurately reconstruct the vector from the trace. For recognition, we only need to store enough information to distinguish a vector from other possible vectors. If we consider that we have a set of $M$ equiprobable $n$-length vectors, then we only require $2k \log_2 M$ bits of information to represent $k$ pairs of those items for recognition.

Circular convolution is efficiently computed in time $O(n \log n)$ using the Fast Fourier Transform to transform the vectors into the frequency domain and performing element-wise multiplication to obtain the resulting vector. This is in contrast to the $O(n^2)$ time needed to compute circular convolution directly from Equation 4.1. In fact, Plate [39] shows that HRRs can be constructed and operated on entirely within the Fast Fourier Transform frequency domain, removing the requirement to transform the vectors back and forth from this space.

### 4.2.2 HRRs as a Superposition Memory

A simple way to store multiple vectors in a single representation is to use superposition to simply add the vectors together. This method for combining vectors does not allow for the exact reconstruction or recall of the constituent vectors, but it does allow for recognition of the constituent vectors being included in the vector sum, under the right conditions. Superposition memories are useful since if vectors with randomly and independently distributed elements are added together, the resulting vector will be more similar to the constituent vectors than to any other vector. When superposition is applied to convolution based memories, like HRRs, this property continues to hold, allowing for multiple convolution traces to be added into a single representation.

Anderson [2] has shown that it is simple to analyze the probability of recognizing a random vector stored within a superposition memory. Plate [39] shows that this analysis holds for convolution memories (i.e. HRRs) and he is able to determine an expression for the probability of recognizing all of the vectors contained within a trace from a given vector set. Plate calls this probability Pr(All Correct) and shows that it can be calculated for $n$-length vectors as

$$\Pr(\text{All Correct}) = \max_t \Pr(s_a > t)^k \Pr(s_r < t)^{m-k}, \tag{4.3}$$

where $k$ is the number of vectors in the superposition, $m$ is the total number of vectors in our vector set, $s_a$ is the accept signal defined as $s_a = N(1, (k+1)/n)$, and $s_r$ is the reject signal defined as $s_r = N(0, k/n)$. In [39], Plate provides a result for the threshold value $t$ through numerical optimization of this expression, which we omit for brevity.

### 4.2.3 Mathematical Properties of HRRs

There are two properties of circular convolution that make it ideal to use as a binding operation, which follow when we assume the HRR elements are normally distributed with a mean of 0 and a variance of $1/n$. First, the expected similarity between a convolution and its constituents is zero. For example, Plate [39] has shown that $\underline{a} \cdot \underline{a} \circledast \underline{b}$ falls within the distribution with a mean of 0 and a variance of $2n + 1/n^2$. Also, the dot product between two convolution traces that share the same constituent vector but are convolved to separate independent vectors, $\underline{a} \circledast \underline{b} \cdot \underline{a} \circledast \underline{c}$, will fall within a distribution of mean 0 and a variance of $2n + 2/n^2$.

Second, two vectors with high similarity, each convolved to the same vector, will produce similar convolution traces. Figure 4.2 highlights this property showing HRR dot product similarity as a function of the similarity of the constituent vectors (in the case of this graph, one constituent vector is referred to as the “role” and the other is referred to as the “filler). Looking at the edges of this graph, for the cases when the “role” or “filler” dot product similarity used in two HRRs is 1, we see that the similarity of those two HRRs is nearly equivalent to the similarity of the other constituent vector. Investigating this graph further, for the cases when the “role” or “filler” dot product similarity used in two HRRs is 0, we see that the similarity of those two HRRs will always be nearly 0, regardless of the similarity of the other constituent vector. These properties allow HRRs to either preserve

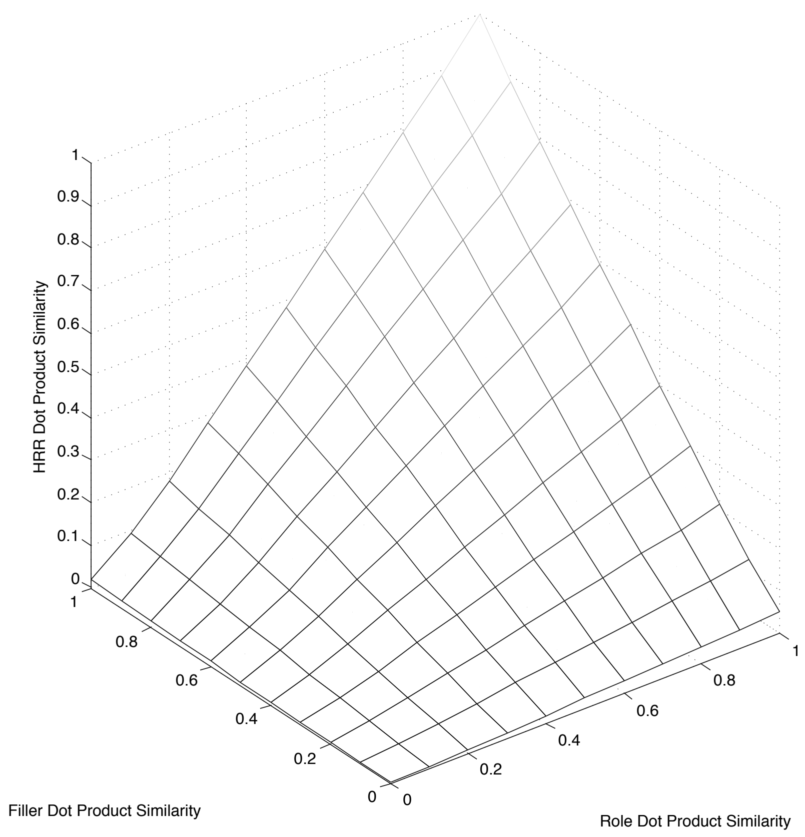


Figure 4.2: HRR Dot Product Similarity as a Function of the Dot Product Similarity of the Constituent "Role" and "Filler" Vectors in 1024 Dimensional Space.

the similarity space of its constituent vectors, by binding those vectors to the same vector, or destroy the similarity space of its constituent vectors, by binding those vectors to orthogonal vectors.

Additionally, and as we have discussed in the previous section, combining HRRs using superposition is another method to combine HRR vectors. Superposition always preserves the similarity of the resulting HRR vector to its constituent vectors. This is in contrast to circular convolution, mentioned above. Armed with these mathematical properties of HRRs, powerful and elaborate structures can be represented within a vector space.

## 4.3 HRR Text Representation

Returning to our discussion in Chapter 2 on text representation techniques, we saw in Section 2.2.6 that including part-of-speech data within a text representation can help model polysemy relationships contained within the text. Unfortunately, no text representation method has been proposed that can include this kind of linguistic information in a way that improves classification scores for text. For example, both Kehagias et al. [26] and Moschitti and Basili [37] create text representation techniques that include PoS type data, but do not produce better classifications scores than traditional text representation methods.

However, we propose that building a text representation scheme using HRRs can help us encode both the semantics and the syntax of a text document in a way that will be useful to computational language processing tasks, such as text classification. It is important to note that this is not the first time that HRRs have been used to create language models. Eliasmith and Thagard [11] have previously shown that HRRs can be used to model both syntactic and semantic psychological data. As well, Eliasmith [10] has shown that HRRs can be successfully applied to model cognitive language processing. However, both of these examples have applied HRRs in the field of cognitive science. Our proposal represents the first time that HRRs have been used to create language representations for machine intelligence applications, such as text classification.

Our natural language representation takes advantage of the ability of HRRs to encode a document's structure in a way that is non-destructive to the document's semantics. Using the circular convolution and superposition operations of HRRs, our representation scheme augments the traditional Bag-of-Concepts semantic model with PoS information to better model polysemy relationships in the representation. The intuition behind this approach, is that we can "bind", using circular convolution, a word's PoS information with a word's term context vector in order to encode both pieces of information in our vector representation.

The properties of HRRs make it an excellent framework to construct a text representation to achieve our goals. First, recall that the expected similarity between a circular convolution and its constituents is zero; therefore, the same term acting as different parts of speech in similar contexts would not be similar in their bound HRR representation. Returning to our commonly used example of the word *can* that can act as both a noun and a verb, the HRR term vector for the word $can_{NOUN}$ in the sentence "He kicked the *can*." would be distinct from the HRR term vector for the word $can_{VERB}$ in the sentence "He *can* kick." More importantly, these different vectors for the word *can* will be approximately orthogonal to each other, as shown in Figure 4.2. Second, the dimensionality of the vectors are constant under HRR operations, so the number of vectors encoded in the structure does not affect the complexity of the representation. This allows us to avoid the curse of dimensionality to which other methods (e.g. Tensor Products [51]) would be susceptible. Third, similar semantic concepts bound to the same PoS identifier result in similar vectors. In fact, as shown in Figure 4.2, the similarity of the bound HRR vector will be approximately equal to the similarity of the original context vector when bound to the same PoS identifier. So, since similarity reflects the structure of the semantic space, these binding operations usefully preserve the relevant geometric relations of the original semantic space. Fourth, since superposition with HRRs returns a vector that is more similar to its constituent, we can build document representations by using superposition to combine the bound HRR vectors of the terms contained within that document to create a document vector that is similar

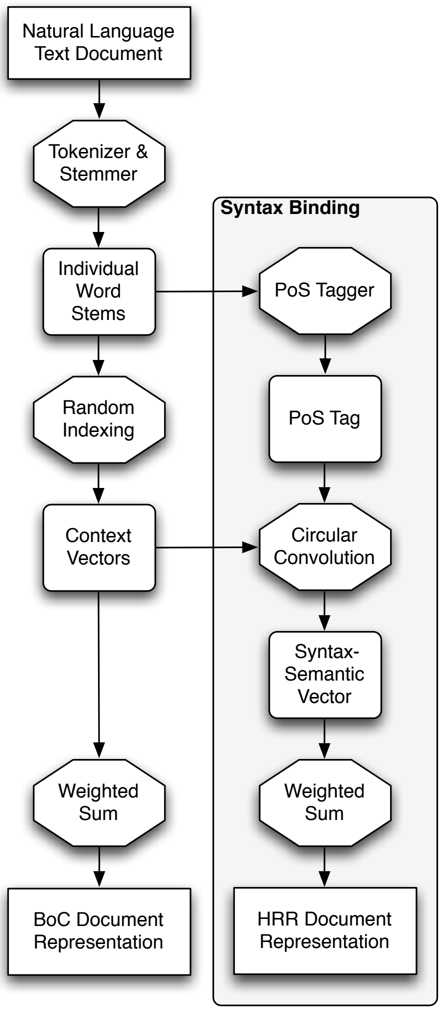


Figure 4.3: Process for Generating BoC and HRR Document Vectors.

to its constituent term vectors.

Figure 4.3 presents the process for generating HRR document representations and the differences in the process from creating BoC document representations. Each octagonal entity in the diagram denotes a processing operation for creating the representations, while each rectangle denotes a data entity that is used as the inputs or represents the output of each processing step. The next few paragraphs describe each processing step in more detail.

Tokenization and stemming act as the first processing step to create both the BoC and the HRR representations. This processing step takes the raw natural language text data as its input. The tokenizer performs simple word tokenization, separating the document's words based on whitespace and punctuation marks found in the document. The stemmer then takes each of these words and performs morphological affix stripping, implemented through simple regular expression matching, to reduce the word to its morphological stem[1]. The result of this tokenization and stemming processing step is an ordered set of individual word stems for the document, which we will denote as the set $S$, that will serve as the inputs to later processing steps.

The next processing step for creating the BoC and HRR Representations is the Random Indexing method (originally described in Section 3.3.3). The random indexer indexes all documents at once in order to generate an accurate model of stem use throughout the text corpora, and creates context vectors for each stem. For the purposes of Figure 4.3, the random indexer takes the ordered set of individual word stems for a document $S$, and for each stem $s_i \in S$, creates a context vector representation for that stem, which we will denote as $\underline{c}_i$. In Random Indexing, each stem is assigned a random label vector $\underline{l}_i$ of the specified dimensionality, with a small number $+1$s and $-1$s randomly distributed about the vector. The context vectors for each stem are created by summing the random labels of the stems that co-occur with our focus stem in a particular context. More formally, $\underline{c}_i = \sum_{i \in D} \underline{l}_i$, where $D$ is a defined context. The context definition, either DOR or TCOR, and vector dimensionality must be specified as a parameters to the random indexer. The algorithms for DOR and TCOR are shown in Algorithm 3.1 and Algorithm 3.2, respectively. The result of the random indexer is a function $\mathcal{R} : S \rightarrow C$, where $C$ is the set of context vectors created from the indexing. Therefore, we will use our indexer to map a specific word stem $s_i$ to its corresponding context vector $\underline{c}_i$.

In order to create and combine syntax and semantic vector representation of a particular word stem, we need the syntax information of that word stem in a vector form. This is accomplished through our PoS Tagger processing step and is only necessary when creating HRR representations. For each document, the PoS tagger identifies that part-of-speech tag associated with each word in a document. For the purposes of Figure 4.3, the PoS Tagger takes the ordered set of individual word stems for a document $S$, and for each stem $s \in S$, identifies the corresponding tag vector $\underline{t}_i$ from a tag set $T$ of the stem's part-of-speech. Tag vectors are created by assigning a randomly generated vector of the same dimensionality as the context vectors to each tag in the tag set. The result of the PoS tagger is a function $\Gamma : S \rightarrow T$. Therefore, we will use our PoS tagger to map a specific word stem $s_i$ to its corresponding tag vector $\underline{t}_i$.

Our HRR text representation relies on the ability to bind a word's semantic information, which we represent by its context vector $\underline{c}_i$, to its syntactic information, which we represent by its tag vector $\underline{t}_i$, to obtain a single vector that contains the information of both of these elements. To accomplish this we use the circular convolution processing steps. To obtain a single syntax-semantic vector for a word stem $s_i$, which we will denote as $\underline{e}_i$, we circularly convolve the stem's context vector and tag vector together to obtain $\underline{e}_i = \underline{c}_i \otimes \underline{t}_i$.

Finally, for our final document representation we create an appropriate weighted sum for the representation. The weighting factor is the same for both BoC and HRR

[1]In our classification system, we used the Porter Stemmer as our stemming algorithm.

representations. For each word stem $s_i$ we create a weighting factor $w_i = tf_i \times idf_i$, where $tf_i$ denotes the stem's term frequency defined in Equation 2.2 and $idf_i$ denotes the stem's inverse document frequency defined in Equation 2.3. The actual sums for the BoC and HRR are created differently corresponding to the different vectors used in the sums. For the BoC, case we sum each $\underline{c}_i * w_i$ for each $\underline{c}_i$ corresponding to a word stem in the document. Alternatively for the HRR case, we sum each $\underline{e}_i * w_i$ for each $\underline{e}_i$ corresponding to a word stem in the document.

Like BoC document vectors, these HRR document vectors are normalized by dividing by the number of terms in the document in order to ensure that there is no classification bias towards longer documents. But unlike BoC vectors, these HRR document vectors encode both semantic and syntactic information. This process is outlined in pseudocode in Algorithm 4.1.

**Algorithm 4.1** HRR Representation Document Generation

```
for all documents $d \in \mathcal{D}$ do
    $representedDocument \leftarrow$ CreateEmptyVector($k$) {Create an empty $k$-dimensional vector}
    $wordTagMap \leftarrow$ PoSTagger($d$)
    for all term occurrences $t \in d$ of term $t \in \mathcal{T}$ do
        $tag \leftarrow$ GetTagForTerm($t, wordTagMap$)
        $\underline{t}_{tag} \leftarrow$ GetTagVector($tag$) {Returns the $k$-dimensional tag vector}
        $\underline{t}_{context} \leftarrow$ GetContextVector($t$) {Returns the $k$-dimensional context vector for term $t$}
        $\underline{t}_{HRR} \leftarrow$ CircularConvolution($\underline{t}_{context}, \underline{t}_{tag}$)
        $\underline{t}_{HRR} \leftarrow \underline{t}_{HRR} *$ IdfWeighting($t$)
        $representedDocument \leftarrow representedDocument + \underline{t}_{HRR}$
    end for
    $representedDocument \leftarrow$ Normalize($representedDocument$) {Normalize the document for word length}
    Store($representedDocument$)
end for
```

# Chapter 5

# Support Vector Machine Classification

Having previously created our desired representations for our text data, we are now ready to classify these representations using a chosen classification method. In this chapter we give a description of the Support Vector Machine classifier. We start the chapter by giving the mathematical formulation of the learning situation for classification. Next, we describe the concept of separating hyperplanes for classification and present a simple training algorithm for learning these hyperplanes. We conclude the chapter by describing the concept of structural risk minimization and how this concept is implemented to create Support Vector Machines.

## 5.1 Mathematical Formulation of Classifier Learning

The classification problem that is addressed in this thesis is as follows: given a set of labelled documents, how should we label new, unseen documents. Sebastiani [49] calls this a *knowledge free* problem, since, in most cases, there is no extrinsic knowledge or formalized meta-data available to aid in the classification problem. Since we only have superficial definitions of the classes expressed, through document label pairs, we must perform some kind of inductive learning to define formally which documents belong in which class, and to determine classification rules. Luckily, this problem is heavily studied in the machine learning research community.

Most machine learning approaches to multi-label text classification treat the classification problem as consisting of $|\mathcal{C}|$ binary classification problems. Usually classes can be assumed to be disjoint from one another. However, if a class hierarchy is available, class dependence can be determined to aid in classification accuracy and efficiency. If a class hierarchy is known, this is clearly extrinsic knowledge of the classification problem and thus the problem cannot be considered knowledge free.

For the remainder of the thesis, we focus on knowledge free classification problems, and therefore make the assumption that each class is independent.

As originally presented in Section 1.1, the goal of classification learning is to approximate the classification function $\check{\Phi}_h : \mathcal{D} \times \mathcal{C} \mapsto \{True, False\}$, where $\mathcal{D}$ is the set of documents, and $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ is the set of predefined classes. To extend this definition to our multi-label binary classification, we wish to determine the functions $\Phi_i^b : \mathcal{D} \mapsto \{True, False\}$ for all $i \in \{1, \ldots, |\mathcal{C}|\}$. Unfortunately, most machine learning algorithms cannot work directly with text documents and assume that each binary classification problem is under the set $\{\pm 1\}$. Therefore, we will remap our documents as $\mathcal{D} \times \mathcal{C} \mapsto \mathcal{X} \times \{\pm 1\}^{|\mathcal{C}|}$, where $\mathcal{X}$ is the $|\mathcal{F}|$-dimensional feature space. Using this representation, rather than trying to determine the functions $\Phi_i^b$, we attempt to determine the functions $h_i : \mathcal{X} \mapsto \{\pm 1\}$ for all $i \in \{1, \ldots, |\mathcal{C}|\}$. The only constraint on the $h_i$ functions is that they are able to separate our feature space into regions, since we are working with a vector space model. To ease notational constraints for the remainder of this chapter, we will omit references to the $|\mathcal{C}|$ independent classifiers and only focus on a single binary classification problem. We can do this since, by our independence assumption on classes, all conclusions reached by our binary classifier set can transfer directly to the multi-label case.

For the remainder of this thesis, we denote vector representations $\underline{x}_i \in \mathcal{X}$ of documents $d_i \in \mathcal{D}$ as instances. $y_i \in \{\pm 1\}$ will denote the label of the instance $\underline{x}_i$. The $(\underline{x}_i, y_i)$ pairs of instances and labels will be referred to as examples. We will use $\Omega$ to denote the set of preclassified examples. The inductive hypothesis $h$ created from $\Omega$ will be referred to as a learner, and the set of hypotheses $\mathcal{H}$ over a specific feature space $\mathcal{X}$ that the learner is allowed to work with will be called the hypothesis space.

Since this is an inductive learning situation, where the learner is forced to learn only from examples, a *supervised learning* environment will be used. Many different supervised machine learning techniques for text classification exist, such as Naive Bayes, $k$-Nearest Neighbours, Decision Trees, and Artificial Neural Networks. In this thesis we focus on Support Vector Machines (SVMs). The rationale for focusing on SVMs is that they have proven to be extremely effective in text classification literature [17, 57], they are efficient algorithms for classification, they are well motivated by formal computational learning theory, and they examine the entire feature space in order to infer a learning rule.

## 5.2 Separating Hyperplanes

SVM learning is motivated by the concept of *separating hyperplanes*; therefore, it is worth exploring this concept and illustrating a simple algorithm for finding these hyperplanes. Suppose we are given a set of examples $(X, Y)$, where $\underline{x} \in \mathbb{R}^2$ and $y_i \in \{\pm 1\}$, we can consider that the examples are labeled points in a plane. We can say that in this case, under a certain labelling, a separating hyperplane will be a

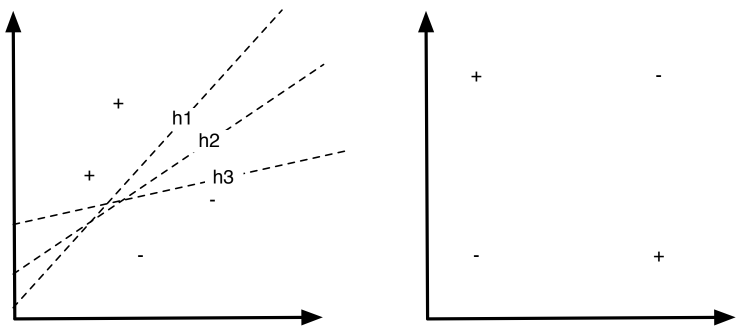


Figure 5.1: Linearly Separable (left) and Non-linearly Separable (right) Examples in 2D Space.

line that divides the plane into two regions, with each region containing only points labeled +1 or only points labeled −1, as in the left of Figure 5.1. Generalizing to $\mathbb{R}^n$, the situation is similar, with our separating hyperplane having a dimensionality of $n-1$. These examples can be considered *linearly separable* under that labeling due to the existence of the separating hyperplane. However, not all sets of examples are linearly separable, as demonstrated by the XOR problem illustrated in the right of Figure 5.1. In the case of the XOR problem, no separating hyperplane can be found given the data.

More formally, if $\mathcal{X}$ is an inner product space we can define a separating hyperplane as

$$h(\underline{x}) = sign(\underline{w}^\top \underline{x} + b) = \begin{cases} +1 & \text{if } \underline{w}^\top \underline{x} + b > 0 \\ -1 & \text{otherwise} \end{cases}, \tag{5.1}$$

where $\underline{x} \in \mathcal{X}$ is an instance, $\underline{w} \in \mathcal{X}$ is the weight vector perpendicular to the hyperplane, and $b \in \mathbb{R}$ is the basis [17]. $b$ defines the location of the hyperplane (when $b = 0$ the hyperplane passes through the origin) and $\underline{w}$ defines the direction of the hyperplane.

## 5.2.1 Perceptron Learning

One of the simplest algorithms for finding a separating hyperplane is the *perceptron learning algorithm*, first defined by Rosenblatt [41]. Minsky and Papert [36] later proved that the algorithm will converge on a separating hyperplane, given that one exists for the problem.

For the case that $b = 0$, the algorithm initializes $\underline{w}$ to a random vector in $\mathcal{X}$ and iterates over each training example $(\underline{x_i}, y_i) \in \Omega$ updating $\underline{w}$ by

$$\underline{w}_{i+1} = \underline{w}_i + \eta(y_i - f_i(\underline{x_i}))\underline{x}_i, \tag{5.2}$$

where $\underline{w}_i$ is the weight vector at iteration $i$, $\eta$ is the learning rate, and $f_i(\cdot)$ is the output of Equation 5.1 at iteration $i$. This equation is commonly referred to as the

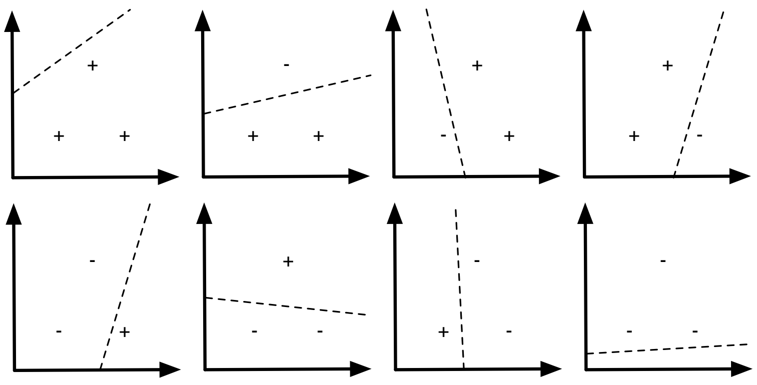


Figure 5.2: Illustration of VC-Theory Guaranteeing Linear Separability for 3 Instances in 2D Space.

perceptron learning rule. Notice that each time $f_i(\underline{x})$ misclassifies the instance $\underline{x}_i$, the weight vector is adjusted to either push it towards or away from $\underline{x}_i$, depending on how the instance was misclassified.

Although the perceptron learning algorithm will converge to a separating hyperplane if one exists, it is unable to handle cases where the examples are not linearly separable. As well, perceptron learning tends to generalize badly for learning, having limited performance on examples not used for training. SVMs will expand on the perceptron concept to overcome these limitations.

### 5.2.2 Vapnik-Chervonenkis Dimension

We have discussed the problem that a given set of examples may not be linearly separable. Vapnik-Chervonenkis (VC) theory [54] generalizes this problem by determining whether a set of instances can be linearly separable irrespective of their labeling.

For the sake of brevity, we will not explore all aspects of VC-theory, but we introduce the fundamental concept of a *VC-dimension* of a hypothesis space. Given a set of instances, if there is a hypothesis space $\mathcal{H}$ and a set of separating hyperplanes that can separate the instances irrespective of any labeling on the instances, we say that the hypothesis space *shatters* that set of instances. From this definition, we can define the VC-dimension of the hypothesis space $\mathcal{H}$, $VC(\mathcal{H})$, as the cardinality of the largest finite subset of $\mathcal{X}$ shattered by $\mathcal{H}$ [54].

Using VC-dimensions, we can now answer the question of whether a set of instances is linearly separable regardless of their labeling. If the hypothesis space $\mathcal{H}$ consists of separating hyperplanes defined over $\mathcal{X} = \mathbb{R}^n$, then we can prove with VC-theory that $VC(\mathcal{H}) = n + 1$. This means that arbitrary subsets of $n + 1$ instances in $\mathbb{R}^n$ are linearly separable. Figure 5.2 illustrates this concept for the $n = 2$ case.

## 5.3 Support Vector Machines

We can use the concept of the VC-dimension and its relationship with the dimensionality of our feature space to conclude that, given a feature space of sufficiently high dimensionality, subsets of this feature space will be linearly separable. SVMs are based on exploiting this finding, but they also try to minimize risk of misclassification associated with inductive learning classifiers.

### 5.3.1 Structural Risk Minimization

One form of risk associated with inductive learning techniques is *empirical risk*, which measure the performance according to some loss metric over the training examples. We can define this type of risk as

$$R_{emp}(h) = \frac{1}{m}\sum_{i=1}^{m} L(h(\underline{x}_i), y_i), \tag{5.3}$$

where $L(\cdot,\cdot)$ is some loss metric function and $m$ is the number of examples [7]. For the remainder of this chapter, we'll focus on the zero-one loss metric [17] due its straightforward analysis, which is defined as

$$L_{0/1}(h(\underline{x}_i, y_i)) = \frac{1}{2}|h(\underline{x}_i) - y_i| = \begin{cases} 0 & \text{if } h(\underline{x}_i) = y_i \\ 1 & \text{otherwise} \end{cases}. \tag{5.4}$$

Notice that all the zero-one loss function does is determines whether the instance was correctly classified.

The second kind of risk associated with inductive learning with which we concern ourselves is called *expected risk* (also called generalization error). Expected risk is defined as the expected loss averaged over unseen examples drawn from the underlying distribution $\Pr(\underline{x}, y)$. More formally, we can define this as [7]

$$R_{exp}(h) = \int L(h(\underline{x}), y)\ \mathrm{d}\Pr(\underline{x}, y). \tag{5.5}$$

Although this kind of risk is difficult to calculate, we can estimate it by sampling the underlying distribution and counting the loss measured by the loss function [7]. This turns the integral in Equation 5.5 into a weighted sum of the loss function on a set of samples.

The relationship between empirical and expected risk highlights one of the major problems for inductive learning: good performance for one risk measure does not necessarily translate to good performance for the other risk measure. For example, in the case of the perceptron, the empirical risk is much lower than the expected risk since the perceptron converges to a hyperplane that fits all of the training examples, but may not be the best one for the complete learning problem. In general this is a problem of *overfitting*, where the hypothesis space has the capacity to capture more complex relationships in the data than actually exist. Overfitting leads to poor generalization of the learning rule, allowing a classifier to achieve zero loss over the training examples but high loss over unseen data from the learning problem. Through VC-theory, we have seen that the $VC(\mathcal{H})$ of a hypothesis space $\mathcal{H}$ can be taken as a measure of the expressiveness of the hypothesis space, and by decreasing this expressiveness measure, we should be able to improve generalization.

Statistical learning theory shows that we can bound the expected risk by the sum of the empirical risk and a hypothesis space capacity term, leading to the relationship

$$R(h) \leq R_{emp}(h) + O\left(\frac{VC(\mathcal{H})\ln\left(\frac{n}{VC(\mathcal{H})}\right) - \ln(\eta)}{n}\right), \tag{5.6}$$

where $n$ is the number of training example used to measure $R_{emp}(h)$, and the inequality will hold for a probability at least $1 - \eta$ [17]. This relationship states that if we wish to keep our expected risk low, we must find a VC-dimension of the hypothesis space that is sufficiently high to capture the complexity of the feature space, but also low enough to avoid overfitting. We call this strategy *structural risk minimization.*

In order for structural risk minimization to be of any use in determining separating hyperplanes, we must find a way to control the VC-dimension of our hypothesis space. Fortunately, we can use the concept of *margins* to accomplish this. In terms of separating hyperplanes, the margin $\delta$ is defined as the distance between the hyperplane and its closest instance in the feature space. Vapnik [54] shows that if we order hypothesis spaces from higher to lower according to the width of their margins, $\delta(\mathcal{H}_1) \geq \ldots \geq \delta(\mathcal{H}_i) \geq \ldots$, then this ordering corresponds to a lower to higher ordering of the VC-dimensions of the hypothesis space, $VC(\mathcal{H}_1) \leq \ldots \leq VC(\mathcal{H}_i) \leq \ldots$.

### 5.3.2 Hard Margin SVM

Recall that when we introduced the concept of separating hyperplanes in Section 5.2, we made no mention of the concept of a margin. However, taking the margin into account is a key step in controlling the VC-dimension of our hypothesis space and implementing structural risk minimization in our classifier. We define a *canonical separating hyperplane* as a hyperplane that is constrained by

$$\begin{array}{ll} \underline{w}^\top \underline{x} + b \geq 1 & \text{when } y = 1, \\ \underline{w}^\top \underline{x} + b \leq -1 & \text{when } y = -1. \end{array} \tag{5.7}$$

The equalities will hold in Equation 5.7 for the positive and negative *support vectors*, the examples that lie on the margin of the hyperplane, while the inequalities in the equation enforce that the examples must lie on the correct side of the separating hyperplane. Note that the definition of this hyperplane assumes that the classes are linearly separable. Fulfilling these constraints, we can show that the margin of the hyperplane is

$$\delta = \frac{2}{\underline{w}^\top \underline{w}} = \frac{2}{||\underline{w}||^2}. \tag{5.8}$$

We can now use Equations 5.7 and 5.8 to define the optimization problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||\underline{w}||^2, \\ \text{subject to} & y_i(\underline{w}^\top \underline{x}_i + b) \geq 1, \forall_{i=1}^{l}, \end{array} \tag{5.9}$$

where $l$ is the number of training examples. Practically, we solve this optimization problem by rewriting it in its dual formulation using Lagrange multipliers, which transforms the problem to

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \underline{x}_i^\top \underline{x}_j, \\ \text{subject to} & \alpha_i \geq 0, \\ \text{and} & \sum_{i=1}^{l} \alpha_i y_i = 0, \end{array} \tag{5.10}$$

where the $\alpha_i$s establish the dual form of the weight vector in the primal form defined by the relationship $\underline{w} = \sum_i \alpha_i y_i \underline{x}_i$ and $\alpha_i = 0$ when $\underline{x}$ is a support vector [7].

We can now define our SVM classifier function as

$$h(\underline{x}) = sign(\underline{w}^\top \underline{x} + b) = sign\left(\sum_{i=1}^{l} \alpha_i y_i \underline{x}_i^\top \underline{x}_i + b\right), \tag{5.11}$$

where $b$ can be derived from Equation 5.7.

### 5.3.3 Soft Margin SVM

The derivation for a hard margin SVM in the previous section only holds when the examples are linearly separable. Often, this is not the case with data presented for a learning problem. Recall that our definition of a canonical separating hyperplane in 5.7 assumes that the classes are linearly separable. We now need a way of modifying our definition of a canonical separating hyperplane to handle examples that are not linearly separable. We can accomplish this by introducing slack variables $\xi_i$ and redefining our canonical separating hyperplane as

$$\forall_{i=1}^{l} : y_i(\underline{w}^\top \underline{x}_i + b) \geq 1 - \xi_i, \tag{5.12}$$

where $\xi_i \geq 0$, $\xi_i > 0$ if $(\underline{x}_i, y_i)$ lies inside the margin, and $\xi_i > 1$ if $(\underline{x}_i, y_i)$ lies on the wrong side of the separating hyperplane [7].

If we use this new definition for our canonical separating hyperplane, we can redefine our optimization problem from Equation 5.9 as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||\underline{w}||^2 + C\sum_{i=1}^{l} \xi_i, \\ \text{subject to} & y_i(\underline{w}^\top \underline{x}_i + b) \geq 1 - \xi_i, \forall_{i=1}^{l}, \end{array} \tag{5.13}$$

where $C$ is called the slack parameter and represents a trade off between the size of the margin and the number of classification errors during training. Note how this optimization problem implicitly makes use of the $L_{0/1}$ loss function described in Section 5.3.1. Again, this optimization is more easily solved in its dual form using Lagrange multipliers, especially since the slack variables disappear in the dual form using the $L_{0/1}$ loss function [7].

### 5.3.4 Non-linear Mappings Using Kernels

We have shown that we can extend SVMs by using soft margins to allow for classification of non-linearly separable data. However, soft margin SVMs still use a linear classification function to perform the actual classification. Another technique that is independent of soft margin SVMs is to transform the original feature space non-linearly into another space of high dimensionality. Therefore, rather than working with our feature space $\mathcal{X}$ we will work with the space $\phi(\mathcal{X})$, where $\phi(\cdot)$ is some non-linear mapping function.

Cover's theorem gives us motivation for performing this transformation [7]. If we project our feature space into a sufficiently high dimensional space, we can guarantee linear separability by this theorem. However, the computational complexity of SVMs depend on the dimensionality of our instances; therefore, we would like to be able to perform these calculations in a lower dimensional space. We can get around this problem by using *kernel functions*, functions that allow us to calculate the inner product of vectors from the high dimensional space in the lower dimensional space. Specifically for the SVM example, rather than calculate $\phi(\underline{x}_i)^\top \phi(\underline{x}_j)$ in $\phi(\mathcal{X})$, kernel functions allow us to calculate it in $\mathcal{X}$. When we use kernel functions in our optimization problem, they allow our SVM to find an optimal separating hyperplane in the high dimensional space that corresponds to a non-linear decision function in our original space. In order to use kernel functions, we only have to substitute $\underline{x}_i^\top \underline{x}_j$ in our problem formulation with $\mathcal{K}(\underline{x}_i, \underline{x}_j)$, where $\mathcal{K}(\cdot, \cdot)$ is the desired kernel function.

Table 5.1: Popular SVM Kernel Functions.

| **Kernel Name** | **Equation** |
|---|---|
| Linear | $\mathcal{K}(\underline{x}_i, \underline{x}_j) = \underline{x}_i^\top \underline{x}_j$ |
| Polynomial | $\mathcal{K}(\underline{x}_i, \underline{x}_j) = (\underline{x}_i^\top \underline{x}_j + 1)^d$ |
| Radial Basis | $\mathcal{K}(\underline{x}_i, \underline{x}_j) = e^{\frac{1}{2\sigma^2}\|\|\underline{x}_i - \underline{x}_j\|\|}$ |

Popular kernel functions used in SVMs are highlighted in Table 5.1, where $d$ is the degree of the polynomial kernel, and $\sigma^2$ is the Gaussian variance used in the radial basis kernel. Obviously the linear kernel is not a non-linear kernel function, but it is included to show how SVM formulations generalized for kernel functions can still incorporate linear maps of the original space. In this thesis, we have chosen to use a linear kernel function in our experiments due to its lack of parameters we need to tune.

# Chapter 6

# Classifier Evaluation

Once classification of our representations has been performed, we can evaluate a classifier based on its classification effectiveness; a metric that judges whether the classifier classifies data well. We begin this chapter with a brief discussion of some of the challenges for classifier evaluation, focusing specifically on text classification. We then move to a discussion of data partitioning to determine how best to use the preclassified data available in the classification problem. Next, we introduce various evaluation metrics used in classification. Finally, we conclude the chapter with a discussion highlighting some of the remaining issues in classifier evaluation.

## 6.1 Challenges in Classifier Evaluation

Recall from Section 1.1 that we introduced the different phases of text classification systems, namely document representation, classifier learning and classifier evaluation. Also recall how choices in one phase of the system can have large effects on other system phases. This interdependence between system phases, like in most machine learning problems, makes it impossible to come up with an analytical evaluation of the system, in addition to making it difficult to evaluate the system empirically [49]. As well, the limited amount of preclassified text corpora available compared to the amount of text data leads to high variance in our empirical classifier evaluation measures.

Most importantly, text classification is a very subjective process even in the bounds of human classifiers. As a result, human classifications of text may differ across individuals, making it harder to learn an acceptable automatic text classification rule. We can illustrate this point by looking at the *inter-indexer consistency* of two human indexers $a$ and $b$ defined as

$$IC_{a,b} = \frac{N}{A+B+N}, \tag{6.1}$$

where $A$ is the number of unique classes assigned by one indexer, $B$ is the number of unique classes assigned by the other indexer, and $N$ is the number of unique classes commonly assigned by both indexers. In some domains, inter-indexer consistency has been reported to be as low as 35%-40% [53]. Soergel [52] notes that if the quality of a classification is poor, high consistency is useless in a system. Soergel continues to say that we must be careful to evaluate a classification relative to a specific purpose or goal that we wish to accomplish.

Returning to the technical side of machine learning, all machine learning algorithms assume that the data of the learning problem are drawn from a single, stable underlying distribution. In the case of text classification, we can call this assumption into question since the classification domain for text classification is continuously and infinitely evolving and since human classifiers may have inherent biases that make them classify documents in particular ways. All of these issues make it difficult to define a *gold standard* for text classification. A gold standard is the best possible classification result, to measure other classifications against.

## 6.2 Data Partitioning

Our primary concern for measuring the effectiveness of a classifier is determining how well the classifier can generalize to data within a particular domain, not just the classification performance on the data used to train it. One way this can be accomplished is to use as many preclassified examples as possible for the training data. However, the amount of preclassified data available is often limited and the variance of the measured effectiveness will be high. As well, we also require some unseen preclassified data to evaluate the generalization performance of the classifier. We desire some method that can optimize the use of this available data, and which will help us control the variance of the measured effectiveness.

A simple way to accomplish this is to hold back a certain number of preclassified examples as a test set. We can partition the set $\Omega$ of preclassified examples into a training set $Tr$ and a testing set $Te$. We use the training set $Tr$ to construct our classifier, while the testing set $Te$ is used to evaluate classification effectiveness. It is important to note that we have not optimally used all of our preclassified examples as the examples in $Te$ are not used for learning and are essentially wasted in constructing the classifier.

A method which is more representative of the available data is *cross-validation* (CV) [42]. CV can occur in various forms, such as $k$-fold CV and leave-one-out CV [42]. Note that leave-one-out CV is just a special case of $k$-fold CV where $k = |\Omega|$. For CV, the effectiveness of a classifier $h$ trained by a learning algorithm $\mathcal{L}$ is taken to be the average effectiveness of that classifier trained on different folds of the data. Each fold corresponds to a different partitioning of $\Omega$. Theoretically, leave-one-out CV will result in a better estimate of the effectiveness of the classifier; however, its computational complexity is quite large since $|\Omega|$ classifiers must be trained for each class. As a result, $k$-fold CV is more commonly used, often with choices of $k$ of either $k = 5$ or $k = 10$ for typical $\Omega$. Algorithm 6.1 shows the pseudocode implementing $k$-fold CV.

Recall from Section 5.1 that we have broken down our multi-label classifier problem into several binary classification problems. This breakdown represents an interesting challenge to CV, namely how to select appropriate amount of negative examples for our folds since we have multiple different classes from which to select. Two possible schemes exist to solve this problem [27]. The first, *balanced CV*, uniformly selects negative examples from the negative classes to give the same number of negative examples as positive within the folds. The second, *stratified CV*, selects examples from the negative classes in the same proportion as they appear in the complete data set. Stratified CV is more desirable in text classification since the number of documents belonging to different classes will vary substantially over a given data set and classifier training and testing will better represent the original distribution of data within the data set. However, in our investigations we have found that stratified CV tends to produce much lower classification scores than balanced CV.

**Algorithm 6.1** $k$-fold Cross-Validation

```
for $i \leftarrow 1$ to $k$ do
    $Te_i \leftarrow$ PARTITION($\Omega, k, i$) {Partitions $\Omega$ into $k$ non-overlapping subsets}
    $Tr_i \leftarrow \Omega - Te_i$
    $h_i \leftarrow \mathcal{L}(Tr_i)$ {Learn a classifier using $Tr_i$}
    $l_i \leftarrow$ EVALUATE($h_i, Te_i$) {Evaluates classifier $h_i$ on $Te_i$ using an arbitrary evaluation function}
end for
$l_{avg} \leftarrow \frac{1}{k} \sum_{i=1}^{k} l_i$
```

## 6.3 Evaluation Metrics

Various evaluation metrics have been proposed for classifier evaluation. Here, we introduce some of the most common measures.

Two common metrics for evaluating classifier effectiveness is *accuracy* and *error*. Accuracy is defined as the probability that a random document $d_j \in \mathcal{D}$ belongs to the set $Corr_i$, the correctly classified documents for classifier $i$. More formally,

$$A_i = \Pr(d_j \in Corr_i), \tag{6.2}$$

and we can calculate an estimate for accuracy as

$$\hat{A}_i = \frac{|Corr_i|}{|Te_i|}. \tag{6.3}$$

Error is the dual of accuracy and can be defined as $E_i = 1 - A_i$, representing the probability that a random document is misclassified. The error metric corresponds to the empirical risk of the classifier using the $L_{0/1}$ loss function, as previously described in Section 5.3.1. Unfortunately, these measures do not preform well with multi-label classification due to the imbalance in class sizes between $|c_i|$ and $|\bar{c}_i|$. As an example, a classifier that assigns all documents as not belonging to the class $c_i$ could receive good accuracy and error scores, although in reality it would be a poor classifier [56].

Two common measures used primarily in the information retrieval research community are *precision with respect to* $c_i$, denoted as $p_i$, and *recall with respect to* $c_i$, denoted as $r_i$. Information retrieval researchers use these measures in combination with each other to determine a retrieval system's ability to return relevant documents while leaving out irrelevant documents [34].

Before we begin defining precision and recall in the context of text classification, we must first define some notation. Our classifier can give results that can be placed in one of four possible sets: the true positive $TP_i$ set where the document is correctly classified as belonging to class $c_i$, the false positive set $FP_i$ where the document is incorrectly classified as belonging to class $c_i$, the true negative set $TN_i$ where the document is correctly classified as not belonging to class $c_i$, and a false negative set $FN_i$ where the document is incorrectly classified as not belonging to class $c_i$.

We can define $r_i$ in the context of text classification as the conditional probability that, given a random document $d_j \in \mathcal{D}$ is classified as belonging to class $c_i$, this decision is correct. More formally,

$$r_i = \Pr(\Phi(d_j, c_i) = True | \check{\Phi}(d_j, c_i) = True). \tag{6.4}$$

Conversely, we define $p_i$ as the conditional probability that, given that a random document $d_j \in \mathcal{D}$ should be classified as belonging to class $c_i$, this decision is made. More formally,

$$p_i = \Pr(\check{\Phi}(d_j, c_i) = True | \Phi(d_j, c_i) = True). \tag{6.5}$$

However, practically these probabilities are estimated using

$$\hat{r}_i = \frac{|TP_i|}{|TP_i \cup FP_i|}, \tag{6.6}$$

and

$$\hat{p}_i = \frac{|TP_i|}{|TP_i \cup FN_i|}. \tag{6.7}$$

Note that these measures are metrics for individual classes. We will explore two ways of combining these class specific measure into global measures for the data set, namely *microaveraging* and *macroaveraging* [49]. In microaveraging, each class contributes to the global measure for the data set in proportion to the class's share of positive examples. This is calculated for recall and precision as

$$\check{r}^{\mu} = \frac{\sum_{c_i \in \mathcal{C}} |TP_i|}{\sum_{c_i \in \mathcal{C}} |TP_i \cup FP_i|}, \tag{6.8}$$

and

$$\breve{p}^{\mu} = \frac{\sum_{c_i \in \mathcal{C}} |TP_i|}{\sum_{c_i \in \mathcal{C}} |TP_i \cup FN_i|}. \tag{6.9}$$

Conversely, in macroaveraging all classes contribute equally to the global measure. Macroaveraged precision and recall scores are calculated as

$$\breve{r}^{M} = \frac{\sum_{c_i \in \mathcal{C}} \hat{\pi}_i}{|\mathcal{C}|}, \tag{6.10}$$

and

$$\breve{p}^{M} = \frac{\sum_{c_i \in \mathcal{C}} \hat{\rho}_i}{|\mathcal{C}|}. \tag{6.11}$$

Microaveraging and macroaveraging can yield different global measures for the same classification problem. The differences in these measures are especially apparent when there are large variations in the sizes of classes within a data set. If the classifier performs well on small classes but poorly on large classes, macroaveraging will artificially inflate the global result even though the classifier performs poorly on a large proportion of the data set. However, depending on the purpose of the experiments or the data set, this behaviour may be desired. It is important to note that if classes are relatively equal in size in the data set, there will be relatively little variation between the macroaveraged and the microaveraged measures.

We can combine the precision and recall into a single measure using the $\mathcal{F}_\alpha$ measure class, defined as

$$\mathcal{F}_\alpha = \tfrac{(\alpha^2+1)pr}{\alpha^2 r + p} \quad 0 \leq \alpha \leq \infty, \tag{6.12}$$

where $\alpha$ is a parameter that controls the relative importance of precision and recall in the combined measure. When $\alpha \to 0$, the importance of $\rho \to 0$. Conversely, when $\alpha \to \infty$, the importance of $\pi \to 0$. Setting $\alpha = 1$ yields the $\mathcal{F}_1$-measure, which is a widely used measure of effectiveness in text classification [49]. This measure gives equal importance to precision and recall and is equivalent to the harmonic mean of precision and recall. This measure is calculated as

$$\mathcal{F}_1 = \frac{2pr}{r+p}. \tag{6.13}$$

The $\mathcal{F}_1$ measure is commonly multiplied by 100 when results are presented in the literature and we will adopt this convention when we present our own results in Chapter 8.

## 6.4 Further Challenges in Classifier Evaluation

Yang [56] points out, and we have informally verified through our review of the literature, that many researchers make far reaching claims regarding the superiority of their methods for text classification without ensuring that the experiments under consideration are comparable. Therefore, Yang proposes two methods for classifier comparison. The first, *direct comparison*, states that two classifiers $\Phi'$ and $\Phi''$ can be compared only if they are tested on the same set $\Omega$, by the same researcher, with the same preprocessing and weighting scheme, etc. The second, *indirect comparison*, compares two different classifiers tested on the sets $\Omega'$ and $\Omega''$, but compares the classifiers to one or more baseline classifiers $\Phi_1, \ldots, \Phi_n$ also tested on $\Omega'$ and $\Omega''$. However, Yang is primarily concerned with comparison of classifiers and ignores differences in representations in text classification systems. As the goal of this thesis is to compare different text representations, comparison of classification methods is not of direct importance to us. But, we can use the concepts discussed by Yang to help ensure the comparability of our experiments.

One problem with the evaluation metrics discussed in this chapter is that they treat classification correctness as a binary decision, i.e. a document is correctly classified or it is not. If we were to account for inter-indexer consistency, a specific document could belong, and not belong, to a particular class as different indexers could classify that document differently. Also, in practice and depending on the context for classification, some incorrect classifications are more damaging than others. For example, given a hierarchical class structure, a document that is misclassified to a parent, sibling or child class of the hierarchy is probably a better classification than if that document was classified to a class in a completely different part of the hierarchy. Therefore, it should be possible to define an evaluation metric that can take into account class topology for hierarchical class structures or inter-indexer consistency for flat class structures. The class structures of the experiments in this thesis are all flat.

Finally, none of the evaluation metrics discussed in this chapter take into account how well the system performs to the needs of the system users. Perhaps, some measure of utility is required to evaluate text classification systems. Obviously, this measure would be quite subjective and would not be useful to the experiments in this thesis. However, many information retrieval researchers have realised that purely objective measures can not capture elements of user satisfaction in the measures [6].

# Chapter 7

# Experimental Setup

This chapter describes the experiments performed. We begin by reviewing the step of our text classification system, described in the previous chapters. Next, we describe the text corpora used as data for the experiments, paying close attention to the suitability of these corpora for the methods and purposes of the experiments. We then discuss the preprocessing and weighting schemes that we used in the experiments. Next, we introduce the specific experiments that we performed, describing each experiment's purpose and the differing parameters of the experiments. Finally, we conclude this chapter with a description of the software used in the experiments.

## 7.1 Review of a Text Classification System

In the previous chapters, we have described the processing steps required to build a complete text classification system. We now briefly review these steps to illustrate the classification system we have built for use in the experiments described in this chapter.

1. *Preprocessing and Base Representation*: Discussed in Chapter 2, the first step of our text classification system is preprocessing our text data and convert it to a base representation. Our classification system first performs stemming on each word in the corpus and then weights the according to their $tf \times idf$ score in the corpus. We have used BoC as our base text representation and have experimented with both DOR and TCOR context definitions.

2. *Dimensionality Reduction*: Discussed in Chapter 3, the second step of our text classification system is reducing the dimensionality of our base representations to a manageable size to avoid the curse of dimensionality. Our classification system adopts the Random Indexing method of dimensionality reduction.

3. *Text Representations*: The third step of our text classification system is augmenting our base representation with more linguistic structure data. In this

step we bind a terms base representation context vector with part-of-speech information using our HRR text representation, as discussed in Chapter 4.

4. *Classification*: Discussed in Chapter 5, the fourth step of our text classification system is performing the classification on our text representations using a chosen classification method. Our classification system uses Support Vector Machine with a linear kernel to perform our classification step.

5. *Evaluation*: Finally, the last step of our text classification system is evaluating our classification results, as discussed in Chapter 6. Our text classification system reports on $\mathcal{F}_1$, recall, and precision scores using both macroaveraging and microaveraging.

## 7.2 Text Corpora

Collections of natural language text documents is said to constitute a *corpus*[1]. In the context of natural language processing experiments, corpora should exhibit two specific characteristics: the corpus should be representative of real world language use in some domain, and the corpus should be in a format that is easily machine readable.

In the context of text classification, suitable corpora should have a set of classes of documents in which to classify the documents. As well, a certain amount of documents should be preclassified in order to learn classification rules inductively. Corpora should also be representative in the size of their classes, i.e. the number of documents assigned to a certain class should be proportional to the actual number of documents belonging to that class in that domain. Obviously, these proportions will only be estimates since new documents are constantly being produced, but these estimates should appear to be reasonable and well-motivated.

A number of popular, standardised corpora exist in the text classification research community. In this thesis we focus only on two corpora: the 20 Newsgroups Corpus [30] and the Reuters-21578 Corpus [32]. We have chosen these two corpora because: 1) they are popular in the research community; 2) they are from different domains exhibiting different textual properties; and 3) they show similar suitability for our experiments on text representations.

The 20 Newsgroups corpus[2] is a collection of about 20,000 newsgroup postings from the year 1993 distributed, almost evenly, over 20 different newsgroups, each corresponding to a different topic. A document's class assignment was self-classified by the authors when they wrote their posts. Some of the topics in the 20 Newsgroups corpus are related (e.g. `rec.sport.baseball` and `rec.sport.hockey`), while other topics are unrelated (e.g. `comp.graphics` and `sci.space`).

[1]Latin for body.
[2]Available at http://people.csail.mit.edu/jrennie/20Newsgroups.

The Reuters-21578 corpus[3] is a collection of 21,578 financial articles that appeared over the Reuters newswire in 1987, distributed over 91 categories (including an 'unknown' category). Each document was manually classified in this corpus by the newswire's editors. The number of documents assigned to each category varies, giving some categories, like `acq`, a large amount of examples from which to infer a classification rule, while leaving other categories, like `rye`, with fewer examples from which to learn.

Recall that from Section 1.2, the hypothesis of this thesis is that a text representation that is based upon linguistic structure will outperform representation techniques that are more ad-hoc with regards to linguistic theory. In Section 4.3, we proposed a new text representation that incorporates the semantics of a word with its particular use within language, i.e. its syntax. This representation relies on using a word's PoS information to disambiguate words that can be used in multiple parts-of-speech. A natural question that we must answer before we proceed with our experiments is how often do words occur in different parts-of-speech in our selected corpora? If words are only rarely used in different parts-of-speech, then the extra information contained within our representation will be mostly redundant and will be unlikely to increase classification scores compared to traditional representations. However, if a sizable portion of the words in a corpus are used with differing parts-of-speech, our text representation will be well designed to improve classification scores in comparison to tradition representations.

After performing a simple corpus analysis on both corpora, we see that around 10% of the unique word stems from both corpora are assigned to different PoS tags. Figure 7.1 shows the detailed breakdown of the percentage of unique word stems in each corpus that are assigned to multiple PoS tags. This amounts to 11,274 unique word stems in the 20 Newsgroups corpus and 3,757 unique word stems in the Reuters-21578 corpus. This leads us to suspect that since our HRR document representation will preserve the differences in meaning between identical word stems assigned to different PoS tags, our representation will have a classification advantage when compared to traditional representation approaches.

## 7.3 Preprocessing and Weighting

As noted in Section 2.4, we will not use stop word removal in our representation experiments because of its lack of theoretical motivation. There are two related reasons for removing stop words during preprocessing: 1) reducing the computational constraints of the processing system; and 2) removing words that contribute little meaning to the documents. We reject the first reason on the grounds that, empirically, our experiments are not significantly hindered with respect to computational resources by leaving stop words in the corpus. We reject the second reason

[3]Available at http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.

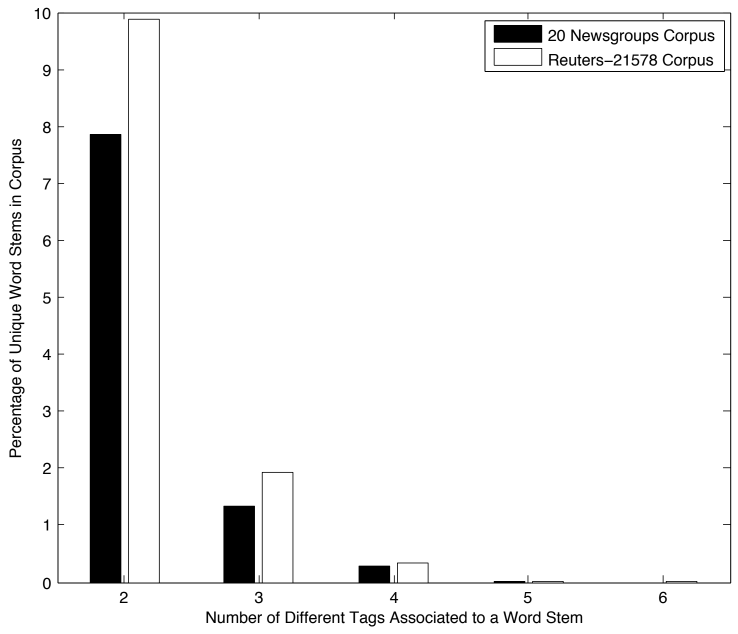


Figure 7.1: Percentage of Unique Word Stems in Each Corpus Assigned to Different Numbers of Part-of-Speech Tags

on the grounds that we have not found a convincing argument in the literature supporting it. On a cursory glance, stop words seem to be equally distributed amongst documents and seem inconsequential to the semantics of a document. However, we suspect that there is indeed a small amount of semantic information contained within the stop words, especially when incorporating the word's PoS information. More importantly, if stop words contain no information about the semantics of documents, leaving them in the corpus will hurt the performance of our system but not its effectiveness. Therefore, we have chosen to process our corpora with stop words.

However, there are other preprocessing tasks that we wish to perform before we build our document representations. The first is tokenization, where the punctuation and whitespace are removed from the documents after identifying the correct boundaries between words. The tokenized words are then stemmed using the Porter stemmer. Table 7.1 illustrates the reduction of the number of words in each corpus to unique word stems contained within that same corpus. Note that the total size of the processed 20 Newsgroups corpus is 118,673 while the the total size of the processed Reuters-21578 corpus is 30,769. This shows that the language use in these two corpora is very different, with the 20 Newsgroups corpus containing more varied language use than the Reuters-21578 corpus.

We use a $tf \times idf$ weighting scheme, as described in Section 2.3. With document-based representations, like BoW, the terms are weighted after processing. In word-based representations, like BoC and HRR, all the elements of a term context vector are weighted using this scheme.

Table 7.1: Total Number of Corpus Words and Unique Stems After Processing.

| **Text Corpus** | **Total Number of Words** | **Number of Unique Word Stems** |
|---|---:|---:|
| 20 Newsgroups | 5 748 345 | 118 673 |
| Reuters-21578 | 2 477 125 | 30 769 |

When we create TCOR representations using random indexing (see Section 3.3.3) for BoC and HRR representations, we use a sliding window as a context. We define our sliding window to extend 4 terms from both sides from both sides of the focus word. We weight the random labels of the surrounding terms in our window by $2^{(1-d)}$, where $d$ is the distance to the focus word. Conversely, in the DOR representations, the concept vectors are created using unweighted term counts. For both TCOR and DOR, the random labels summed to create the context vectors were generated with +1s and -1s distributed over the dimensionality of the vectors representing about 2% of the dimensionality.

## 7.4 Experiments

There are many different interesting experiments that can be performed when comparing representations. In the interests of time, we have only performed a subset of these experiments. The main purpose of these experiments was to compare the classification effectiveness of HRR text representations with traditional text representations. Specifically, we investigated whether using circular convolution to 'bind' PoS information to semantic information would help with classification. Our goal was not to produce a top score for either corpus compared to other reported scores in the literature. As mentioned in Section 6.4, we have not ensured that our experiments are directly or indirectly comparable with other experiments in the literature. However, we have ensured that our experiments are directly comparable with each other, allowing us to draw conclusions between the representations in our experimental setup.

We have limited ourselves to a Support Vector Machine classifier using the linear kernel. This kernel was chosen because it has no parameters to optimize and since experiments reported in the literature indicate that SVM with the linear kernel performs on par with SVM with non-linear kernels in text classification tasks [57]. The SVM slack parameter, explained in Section 5.3.3, was set to 160. In our investigations with the slack parameter, we found that the higher the slack parameter, the better the classification effectiveness; however, the lower the computational performance. This parameter, 160, was selected to give us high classification effectiveness with reasonable computational performance. Our SVM was set to use a $\mathcal{F}_1$ loss function in its optimization (see Sections 5.3.1 and 6.3), which is calculated as 100 minus the $\mathcal{F}_1$ score in percent and directly optimizes the SVM binary classification rules for the $\mathcal{F}_1$ measure [19, 18].

We have also limited our data partitioning scheme for our classification data sets to 10-fold stratified cross-validation, where applicable. As described in Section 6.2, the advantage of stratified cross-validation is that it creates training and testing sets that better reflect the distribution of data in the original corpora; therefore, limiting the effects of noise in the final results. However, from our review of the literature, stratified cross-validation is not a popular choice amongst text classification researchers as it lowers overall classification scores, thus hindering the comparability of the experiments in this thesis to other studies. The number of folds, 10, was chosen arbitrarily to allow us to examine the variance in each classification fold run, while ensuring that our experiments would complete in a timely fashion. As stated in Section 6.2, 5-fold and 10-fold cross-validation are popular choices in the literature [42]. Where possible, we report our results as learning curves, using various percentages of the corpus as training data. This allows us to investigate how the classification scores for the representations scale with amount of data available.

When PoS tags are required to generate the HRR text representations, we use the Penn Treebank tag set. The Penn Treebank project, created and maintained by natural language processing researchers at the University of Pennsylvania, is a corpus with sentence annotations reflecting the syntactic structure of each sentence [35]. The Penn Treebank project has set many standards in text processing research, such as the naming and identification of PoS tags, which is why we adopt it in this thesis. The Penn Treebank tag set contains 36 PoS tags, ranging from specific types of the standard grammatical parts-of-speech (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions) to symbols and numbers.

We report the results of all our experiments using the $\mathcal{F}_1$ measure, as well as precision and recall scores (see Section 6.3) if interesting trends occur. Since we are dealing with two multi-class corpora, we will report both the macroaveraged and the microaveraged results of the experiments.

We omit error bars on our result graphs to make our result graphs more readable. More importantly, the error bars in the graphs represent the overall variability in the corpora, since certain classes will be easier to classify than others yielding higher scores for the class compared to lower scores for harder classes. This variability is not truly representative of the variability of our representations. However, we do calculate statistical significance scores on the differences between methods to evaluate which method is better than others using the paired $t$-test. For these statistical tests, we set our null hypothesis to be that the difference between the two methods under comparison are zero, while our test hypothesis is that one method yields better results than other. To calculate the significance range, we take the mean difference between the two methods over all the runs on a particular class for all classes, and use the variance in these means to arrive at a final value.

In the following subsections, we introduce the specific experiments and their goals that will be reported upon in the next chapter of this thesis.

### 7.4.1 Syntactic Binding Representations

These experiments follow from our previous work in [13]. In Sections 4.2.1 and 4.3, we introduced circular convolution as a method to incorporate structural information with the semantic models of traditional text representations. However, we have not discussed whether simpler methods of 'binding' are equally or more effective for text classification. In these experiments, we investigate other methods of syntactic binding and compare them to the circular convolution method. We will also compare these representations to standard BoC representations, which will have no syntactic information within the representation. In these experiments, the context vectors will be created using the DOR random indexing approach at 512 dimensions. The syntactic information in these experiments will be limited to a collapsed PoS tag set (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions) from the original Penn Treebank tag set [35]. The representations we investigate are introduced below.

#### Multiplicative Binding

The simplest method that we investigate is multiplicative binding. For each PoS tag in our collapsed set, we generate a unique random vector for the tag of the same dimensionality as the term context vectors. For each term context vector, we perform element-wise multiplication between that term's context vector and its identified PoS tag vector to obtain our combined representation for the term. Document vectors are then created by summing the document's combined term vectors.

#### Circular Convolution

These representations represent our novel proposed HRR text representation, described in Section 4.3.

#### Text-based binding

Text-based binding combines a word with its PoS identifier before the semantic modelling is performed. This is accomplished by concatenating each term's identified PoS tag name with the term's text. Then, the concatenated text is used as the input for Random Indexing to determine the term's context vector. Document vectors are then created by summing the document's term vectors. It is important to note that this is the only representation in this experiment that attempts to bind syntactic information before creating the semantic model from random indexing.

### 7.4.2 Dimensionality Investigation

These experiments investigate how the BoC and HRR representation classification effectiveness scale with the dimensionality of the representation vectors. We explore the representations in a logarithmically increasing dimension set of 32, 64, 128, 256, 512, 1024, 2048, and 4096. In these experiments, the context vectors will be created using the DOR random indexing approach. The syntactic information in these experiments will be limited to a collapsed PoS tag set (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions) from the original Penn Treebank tag set.

### 7.4.3 Part-of-Speech Tag Set Size Investigation

These experiments investigate how the HRR representation classification effectiveness is affected by the size of the PoS set. We explore the representations created with a collapsed PoS tag set (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions) and the whole Penn Treebank tag set. In these experiments, the context vectors will be created using the DOR random indexing approach with 512 dimensions.

### 7.4.4 Context Definition Investigation

These experiments investigate how the definition of the context in random indexing for HRR and BoC representations affects classification effectiveness. We explore the representations created with both document occurrence, DOR, and term co-occurrence, TCOR, representations for both BoC and HRR approaches. We will also compare these representations to standard BoW representations. In these experiments, the context vectors will be created with 1024 dimensions. The syntactic information in these experiments will be limited to a collapsed PoS tag set (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions) from the original Penn Treebank tag set. These experiments expand on previous work that we have reported in [12].

### 7.4.5 Limited Vocabulary Investigation

These experiments investigate how limiting a corpus' vocabulary affects classification performance for the representations. In these experiments, for all representations we will randomly remove various percentages of the number of words in a corpus before we generate the representations to determine how the representations will be affected by the limited information. The vocabulary lists were generated independently from one another; therefore, a word that appears when only a low percentage of the corpus vocabulary was used may be omitted when a higher percentage of corpus vocabulary is used. We will investigate the effects of BoC, HRR and BoW representations. In these experiments, the context vectors will be created using the DOR context with 1024 dimensions for the BoC and HRR representations. The syntactic information in these experiments will be limited to a collapsed PoS tag set (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions) from the original Penn Treebank tag set.

## 7.5 Software Used

Wherever possible, we have used prewritten and freely available software to construct our system to help in reproducing our results. Tokenization and PoS tagging was accomplished using MontyLingua 2.1[4], written in Python. The implementation of the Porter Stemmer used in the experiments came from the Natural Language Toolkit 0.9.2[5], also written in Python. We used the JavaSDM 2.22[6] implementation of random indexing; however, we had to heavily modify the source code of the package in order to correct bugs in the random label generation and add support for creation of low dimensional representations. Finally, we used $SVM^{perf}$ 2.1[7] as our Support Vector Machine implementation. We set $SVM^{perf}$ to use the linear kernel and the $\mathcal{F}_1$ loss function in its optimizations.

[4]Available at http://web.media.mit.edu/ hugo/montylingua/.

[5]Available at http://nltk.org/.

[6]Available at http://www.csc.kth.se/tcs/humanlang/tools.html.

[7]Available at http://svmlight.joachims.org/svm_perf.html.

# Chapter 8

# Results and Discussion

In this chapter, we present the results of our experiments described in Chapter 7. We divide this chapter into sections based on each experiment outlined in Section 7.4. In each section we present both the microaveraged and macroaveraged $\mathcal{F}_1$ results, plus macroaveraged and microaveraged precision and recall results if interesting trends appear for the experiments.

Although we attempt to limit each experiment to investigating a specific parameter independently from the others, it is likely that the interactions between these parameters are not independent. As a result, the results presented for each experiment are difficult to present and discuss. In our discussions of these results, we have tried to infer plausible explanations while being careful not to draw too far reaching conclusions.

## 8.1 Syntactic Binding Representations

Looking at the $\mathcal{F}_1$ scores of the syntactic binding representations for the 20 Newsgroups Corpus, as illustrated in Figure 8.1, circular convolution proves to be the most effective method at combining PoS information with a word-space semantic model. In fact, the circular convolution syntactic binding representations provides a statistically significant improvement over BoC representation in a 99% confidence interval using the $t$-test. It is interesting to note that all the syntactic binding methods investigated produce better $\mathcal{F}_1$ scores than the BoC representation under macroaveraging and microaveraging when 60% or more of the corpus was used for training. This allows us to infer that integrating PoS data with a text representation method is indeed beneficial for classification, compared to just classifying on semantics.

Figure 8.2 and Figure 8.3 show the precision and recall scores, respectively, of the syntactic binding representations for the 20 Newsgroups Corpus. In the precision case, the circular convolution representations provide the best scores, with the multiplicative binding representations a close second, while the text-based

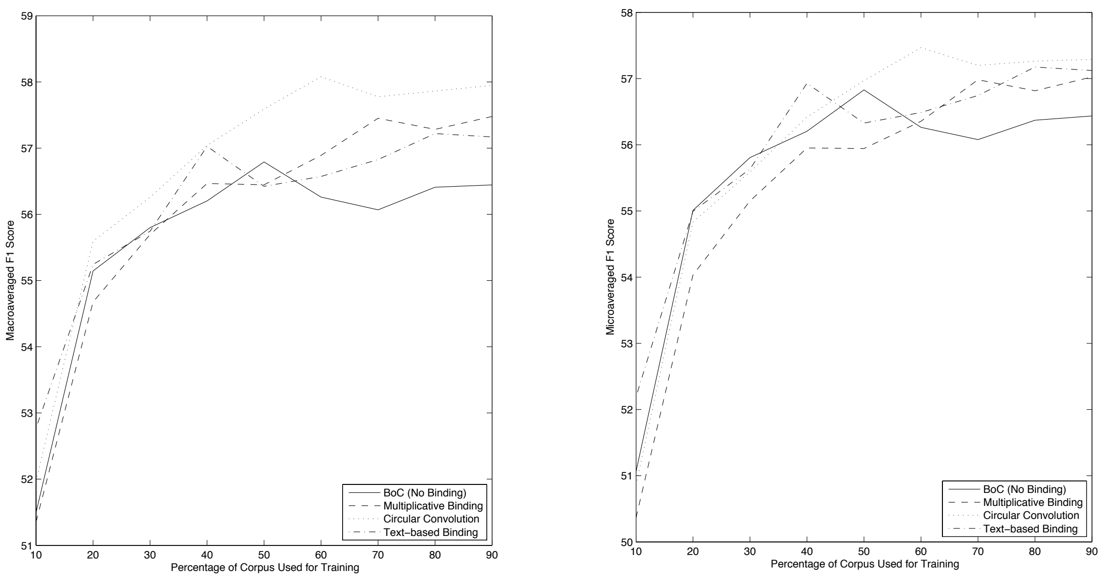


Figure 8.1: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of Syntactic Binding Representations for the 20 Newsgroups Corpus.

binding and BoC representations come in with similar, lower scores. Interestingly, the converse is seen in the recall scores, with a clear ranking of text-based binding, BoC, circular convolution and multiplicative binding.

Recall from Section 2.2.2 that recall scores tend to evaluate the ability of a representation to model synonymy relations, while precision scores tend to model a representation's ability to model polysemy relations. We can interpret these results by explaining that the circular convolution and the multiplicative binding representations do a better job at modelling polysemy, while introducing noise in the synonymy models. Meanwhile, the text-based binding representations and the BoC representations better model synonymy relations at the expense of polysemy. This trade off is important as a theoretically perfect language representation will model both polysemy and synonymy equally well. Because of this, the $\mathcal{F}_1$ scores provide us with a better single evaluation score since the $\mathcal{F}_1$ score weights precision and recall scores equally. Since the circular convolution representations produce the best results under $\mathcal{F}_1$ scoring, we can conclude that the added information used to model polysemy relations outweighs the negative effects of the noise introduced in the synonymy models; therefore, it is the best method for including PoS information for text classification of the methods investigated.

## 8.2 Dimensionality Investigation

Figure 8.4 and Figure 8.5 shows the $\mathcal{F}_1$ scores of the BoC and the HRR representations under various context vector dimensions for the 20 Newsgroups and Reuters-21578 corpora, respectively. For both corpora, the microaveraged $\mathcal{F}_1$ scores for the

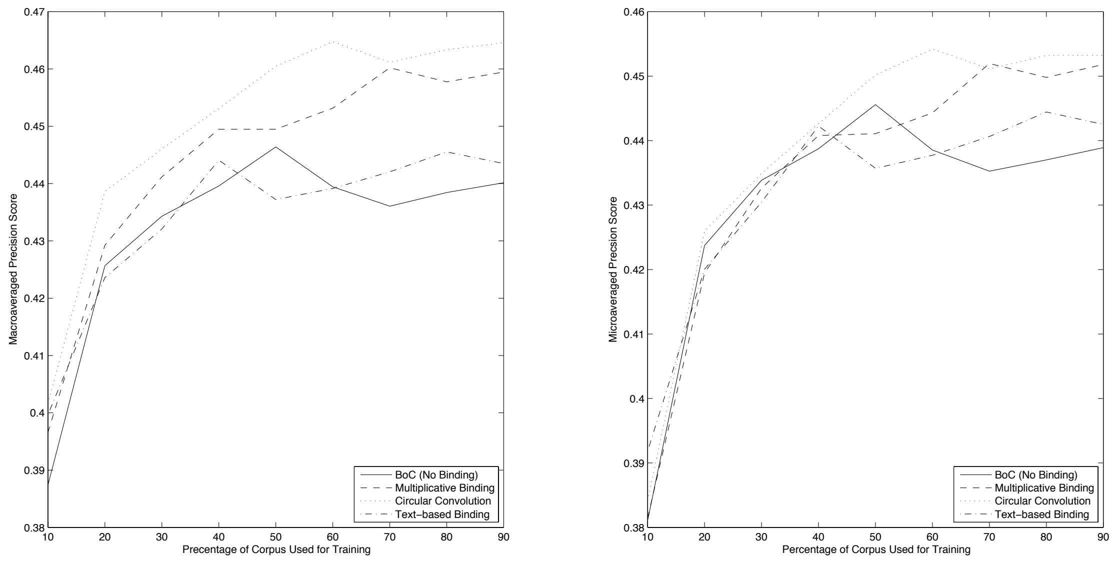


Figure 8.2: Macroaveraged and Microaveraged SVM Precision Scores of Syntactic Binding Representations for the 20 Newsgroups Corpus.

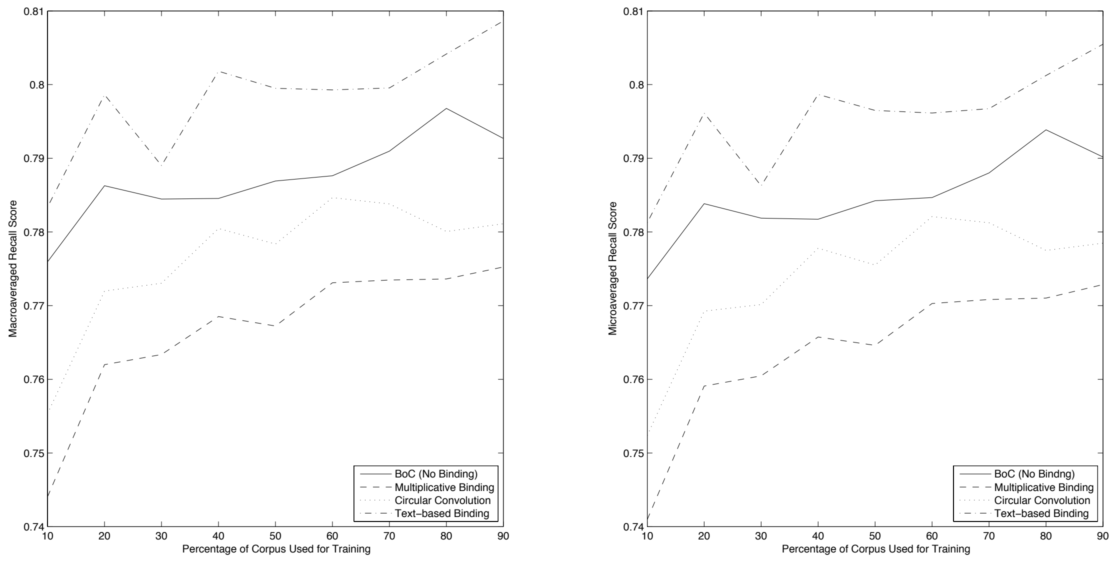


Figure 8.3: Macroaveraged and Microaveraged SVM Recall Scores of Syntactic Binding Representations for the 20 Newsgroups Corpus.

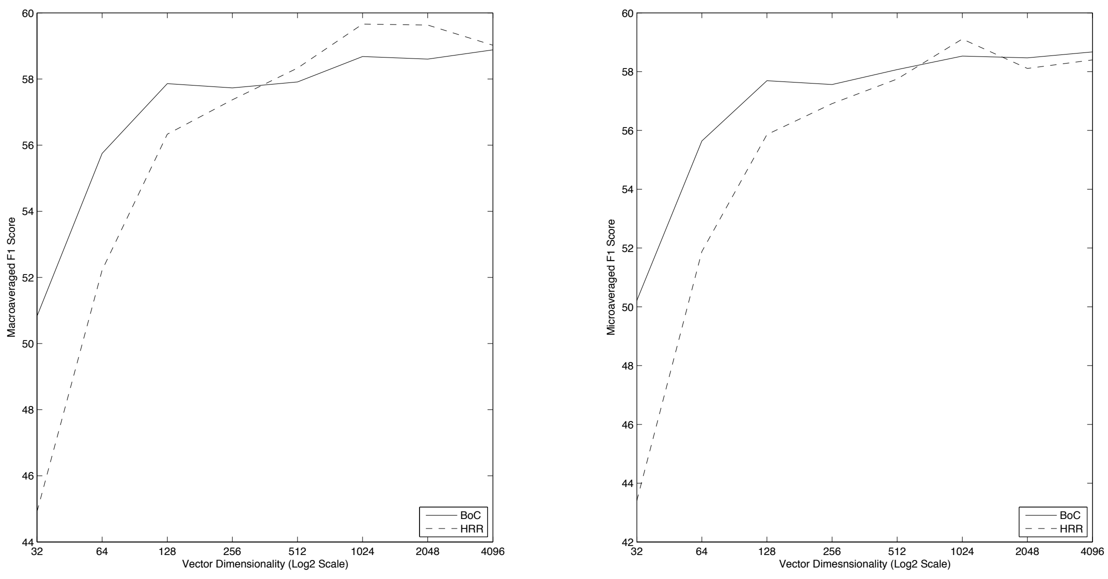


Figure 8.4: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of BoC and HRR Representations Under Varying Dimensionality for the 20 Newsgroups Corpus.

HRR representations seem to be highest at 1024 dimensions. For the Reuters-21578 corpus, both the macroaveraged and microaveraged $\mathcal{F}_1$ scores for the HRR representations show improvements over the scores of the BoC representations after 64 dimensions, while for the 20 Newsgroups corpus these improvements are only seen after 512 dimensions for the macroaveraged case.

Moving our attention to the precision and recall scores for these representations, as illustrated by Figure 8.6 and Figure 8.7 respectively for the 20 Newsgroups and Reuters-21578 corpora, the recall scores show interesting trends as the dimensionality varies. For the 20 Newsgroups corpus, we see that the BoC representations have a higher recall score than the HRR representations, as we expect from the syntactic binding results. However, as the dimensionality increases the difference between the scores decreases due to a greater rise in the recall score of the HRR representations with added dimensions. In the case of the Reuters-21578 corpus, we again see the difference in the recall score shrink between the two representations, but this is due to a decrease in the recall score of the BoC representations rather than an increase in the HRR representations after 128 dimensions. The precision scores for both corpora show similar trends as the $\mathcal{F}_1$ score and have been omitted for brevity.

Interpreting these results is difficult due to the lack of information on the structure of word-spaces created through random indexing (see [24] as an example). We know that when dimensionality is low in random indexing, the technique begins to break down as we will not be able to assign unique random labels to contexts due to the limited number of possibilities over the dimensions. However, the effects of large dimensionality is unknown. These results empirically suggest that higher

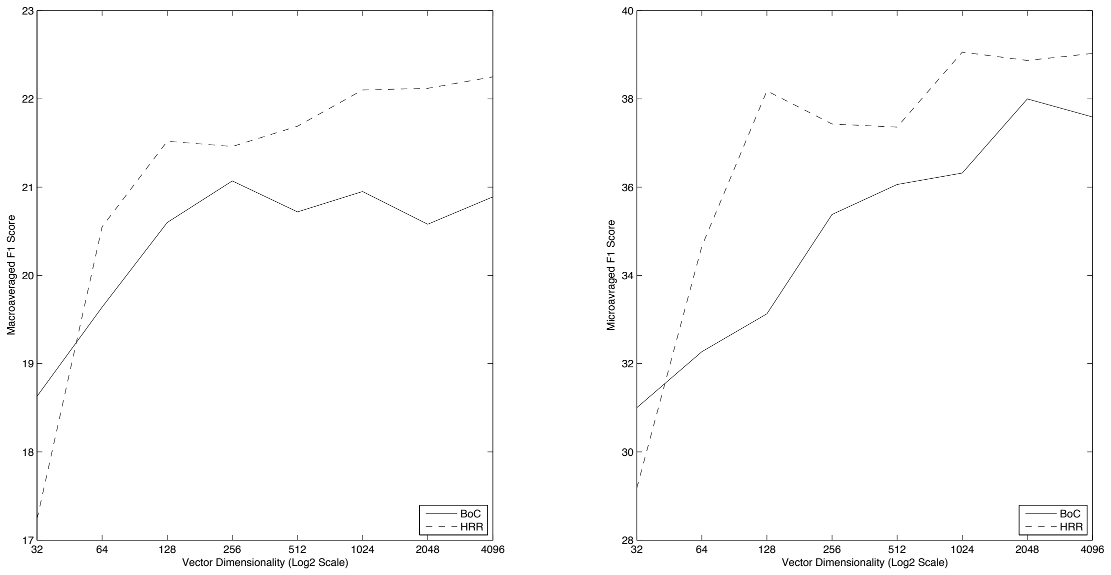


Figure 8.5: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of BoC and HRR Representations Under Varying Dimensionality for the Reuters-21578 Corpus.

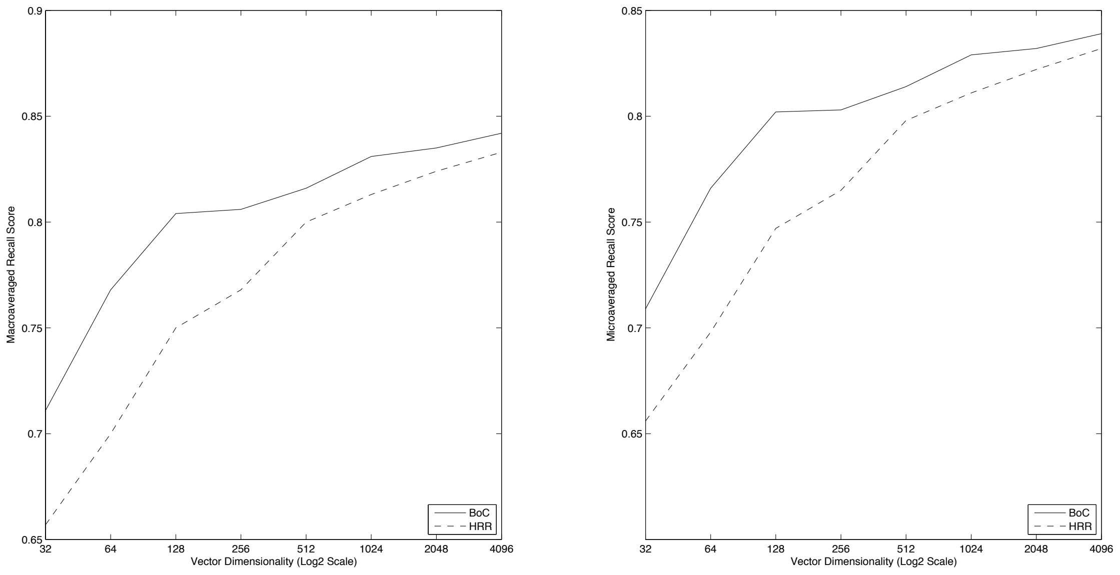


Figure 8.6: Macroaveraged and Microaveraged SVM Recall Scores of BoC and HRR Representations Under Varying Dimensionality for the 20 Newsgroups Corpus.

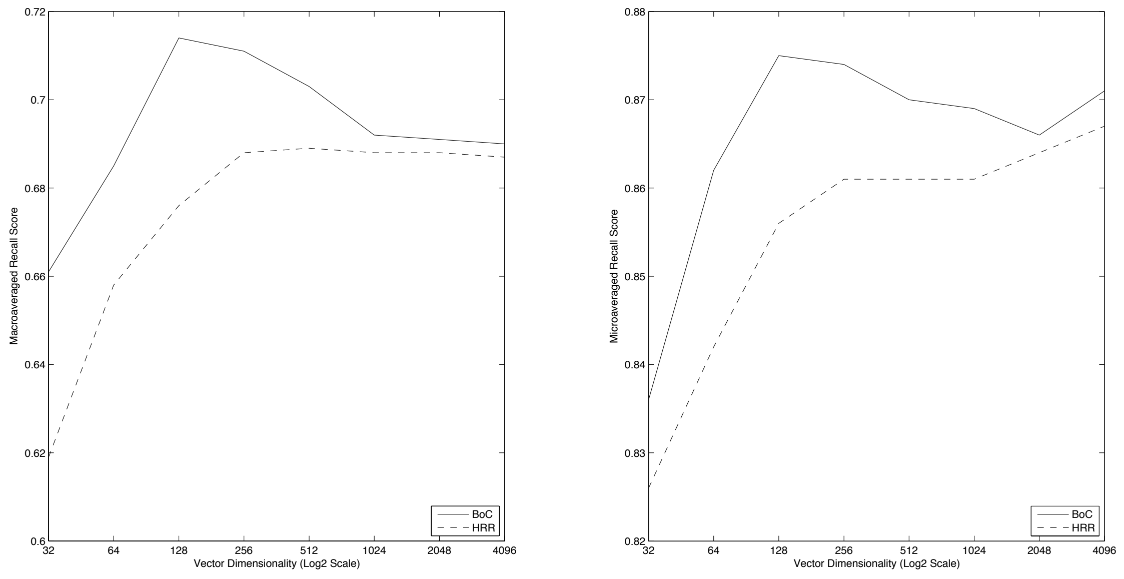


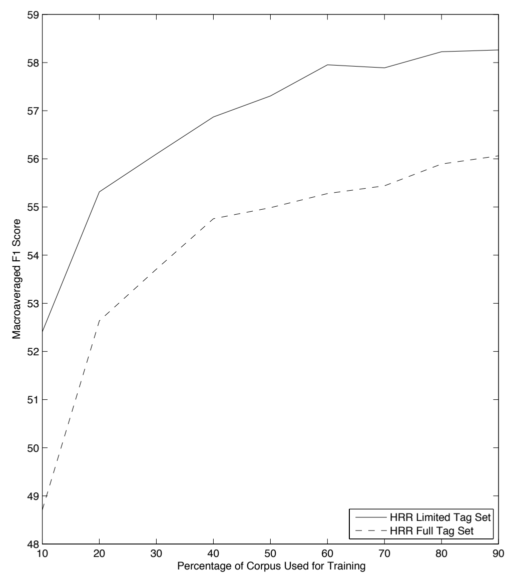
Figure 8.7: Macroaveraged and Microaveraged SVM Recall Scores of BoC and HRR Representations Under Varying Dimensionality for the Reuters-21578 Corpus.

dimensionality in the word-space can significantly effect the recall scores, although having a limited effect on precision scores, of the representations when used in classification. Recalling the results from the syntactic binding representation experiments in the previous section, adding dimensionality to word space seems to be a promising strategy to increase the recall scores of circular convolution based representations, like our HRR text representations.

Interpreting these results more practically for classification system design, these results suggest the selection of 1024 dimensions as the size of our word-space. Choosing our dimensionality to be of this magnitude puts our classification results for the HRR text representations in a suitable performance range. Reconsidering the curse of dimensionality, explained in Section 3.1, 1024 dimensions is a manageable number for our computational methods. For each classification data fold, the SVM classification took about 10 minutes for out HRR representations.

## 8.3 Part-of-Speech Tag Set Size Investigation

The $\mathcal{F}_1$ scores of the HRR representations with varying PoS tag set sizes for the 20 Newsgroups corpus is illustrated in Figure 8.8. These results clearly show that the HRR representations with the limited PoS tag set size, where the tags in the Penn Treebank tag set are collapsed to a simple PoS tag set (i.e. noun, verb, adjective, adverb, interjection, pronoun, preposition, articles, and conjunctions), is superior to using the original tags in the tag set. In fact this superiority is statistically significant under a 99.9% confidence interval using the $t$-test. We omit the precision and recall scores for the corpus since they show the same trends as the



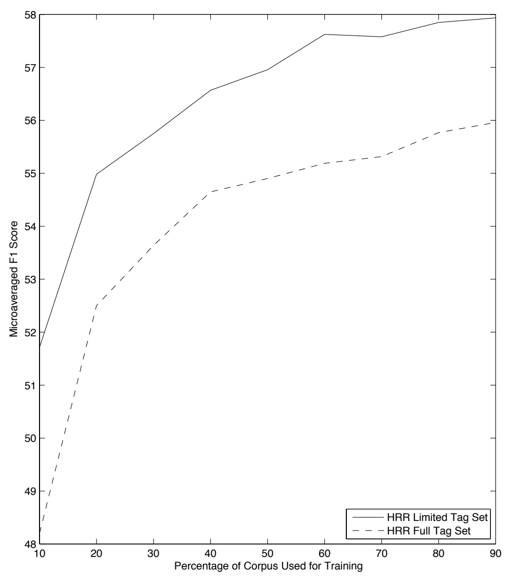
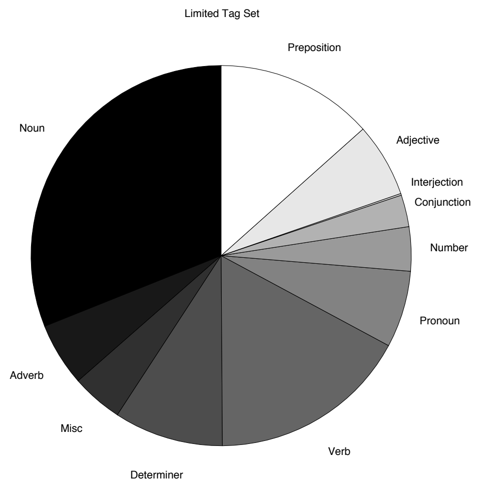


Figure 8.8: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of HRR Representations Created with Varying Tag Set Sizes for the 20 Newsgroups Corpus.

$\mathcal{F}_1$ scores, where the full PoS tag set HRR representations always produce lower scores than the collapsed PoS tag set representations.

This result is seemingly counterintuitive. The complete Penn Treebank tag set has significantly more tags than our collapsed PoS tag set; therefore, including the larger set of information captured by this larger tag set should improve classification scores. However, this is not the case due to a subtle detail of the HRR representations. Figure 8.9 shows the distribution of each tag in the 20 Newsgroups tag set for our limited PoS tag set and the full Penn Treebank tag set[1]. The Penn Treebank tag set has multiple tags for differing parts of speech, while the limited PoS tag set collapses these multiple tags to a single tag. Also, recall from Section 4.3 that the HRR text representations randomly generate each tag vector. In other words, each tag is assumed to be independent from the other tags in the set. Therefore, in the case of the representations created with the full Penn Treebank tag set, each specific tag for the same part-of-speech is treated independently and the desired relationship between the tags is lost in the representations. However, in the limited PoS tag set case, these dependent tags are collapsed to a single tag for that part-of-speech, allowing the representations to correctly model the PoS information for the word and improve classification scores. Collapsing the tag set also removes some tagging errors within the representations created with the collapsed set, leading to more accurate modeling an an increase in performance.

[1]A small number of tag labels have been removed from the distribution graph of the full Penn Treebank tag set to make the graph more readable.



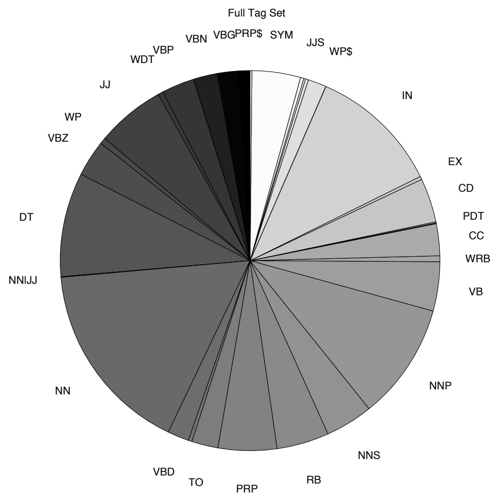


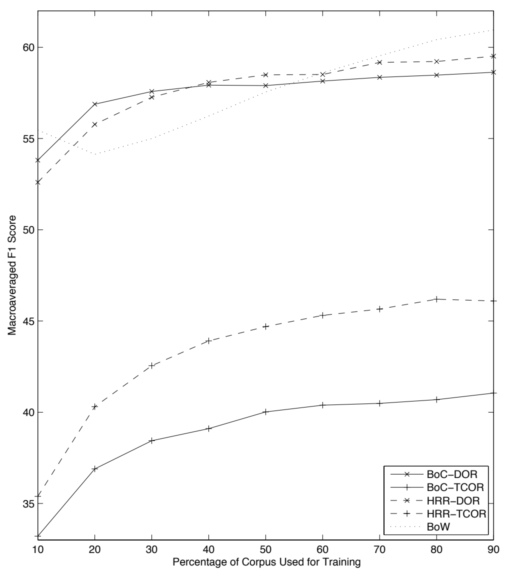
Figure 8.9: Tag Distribution of the Limited and Full Parts-of-Speech Tag Sets for the 20 Newsgroups Corpus.

## 8.4 Context Definition Investigation

Figure 8.10 shows the $\mathcal{F}_1$ scores of both DOR and TCOR generated BoC and HRR representations for the 20 Newsgroups corpus. The $\mathcal{F}_1$ scores of BoW representations are also presented in the figure. The graphs show that the TCOR generated representations performed significantly worse than the DOR representations for both the BoC and HRR representations. These differences are all statistically significant to greater than a 99.9% confidence interval using the $t$-test.

Turning our attention to the DOR generated BoC and HRR representations, in the macroaveraging case we see that the HRR representations yield better classification results when 40% or more of the corpus is used for training, while in the microaveraging case the HRR representations are superior after 60% of the corpus is used for training. Including the BoW representations results in the analysis, we see in the macroaveraging case, the BoW representations yield the best results after 60% of the corpus is used for training, while in the microaveraging case this is true only when 90% of the corpus is used for training. However, in a practical classification situation, it is extremely unlikely that all of the data in the domain will be available to train a classifier; therefore, favouring the HRR representations over the BoW representations. These differences are statistically significant in a 99% confidence interval using the $t$-test.

Investigating the recall and precision scores for this corpus yields some insight into these $\mathcal{F}_1$ scores, as illustrated respectively by Figure 8.11 and Figure 8.12. The recall graph shows that the BoW representations clearly have the advantage in recall scores over all the representations, with BoC-DOR and HRR-DOR as runners up in respective order. More importantly, these recall scores show only a small increase when more training data is added beyond 20% of the corpus. Turning our attention to the precision scores, we see similar trends to the $\mathcal{F}_1$ scores. But what



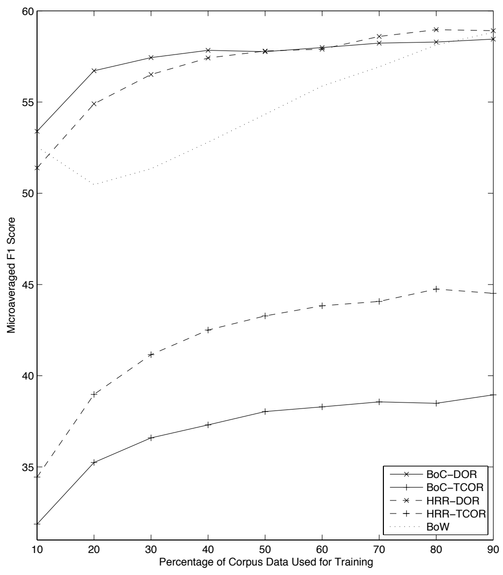


Figure 8.10: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of DOR and TCOR Generated BoC and HRR Representations and BoW representations for the 20 Newsgroups Corpus.

is interesting from the precision scores is that although all representations benefit from adding more data, the BoW representations show the greatest increase. This explains why the BoW representations do so badly in low data cases and better in high data cases: although the representation's recall is high regardless of the amount of data, the precision score is correlated to the amount of data used for training.

We now look at the classification results for the Reuters-21578 corpus. We have omitted the TCOR generated BoC and HRR representations from these results since they continue to preform poorly compared to the DOR representations. The $\mathcal{F}_1$ scores for the BoC, HRR and BoW representations on this corpus are illustrated in 8.13. The graph shows that although the HRR representations beat the BoC representations no matter how much training data is used, the BoW representations always significantly beat the HRR representations. The differences are statistically significant to a 99.9% confidence interval using the $t$-test.

Examining the precision scores for the Reuters-21578 corpus helps to explain the superior $\mathcal{F}_1$ scores of the BoW representations, as illustrated by Figure 8.14. What is interesting about the precision scores is that, unlike the 20 Newsgroups corpus, the BoW representations perform better than the other representations regardless of the amount of data used for training. This precision boost for the BoW representations is the cause of the superior $\mathcal{F}_1$ scores over the BoC and HRR representations, while the BoC and HRR representations show the same precision trends as they did in the 20 Newsgroups corpus. We suspect that the BoW representations are more susceptible to subtle changes between corpora that have a great effect on the classification results of the representations, while the word-space based HRR

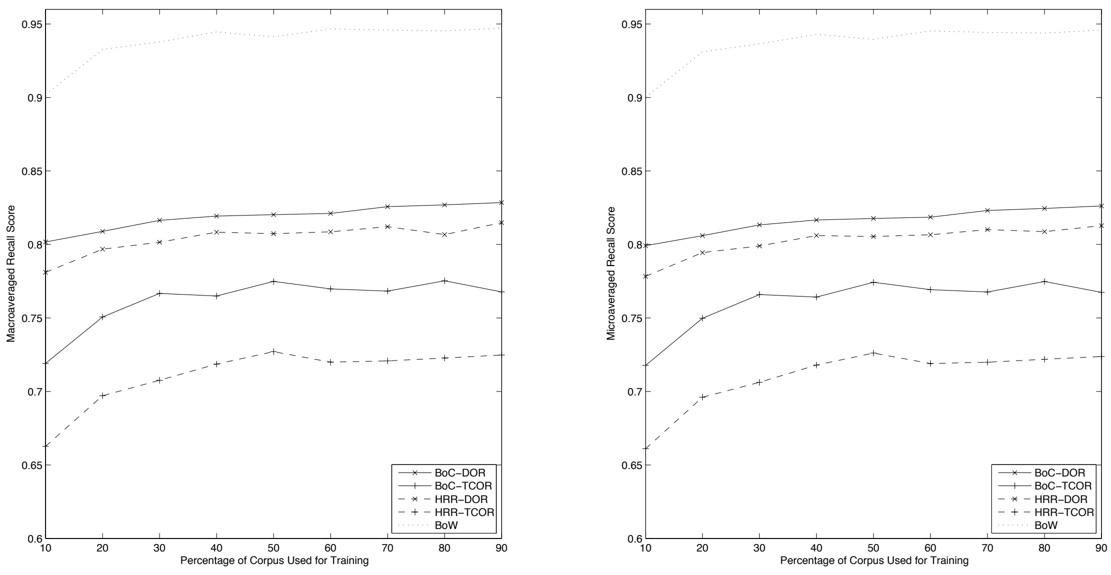


Figure 8.11: Macroaveraged and Microaveraged SVM Recall Scores of DOR and TCOR Generated BoC and HRR Representations and BoW Representations for the 20 Newsgroups Corpus.

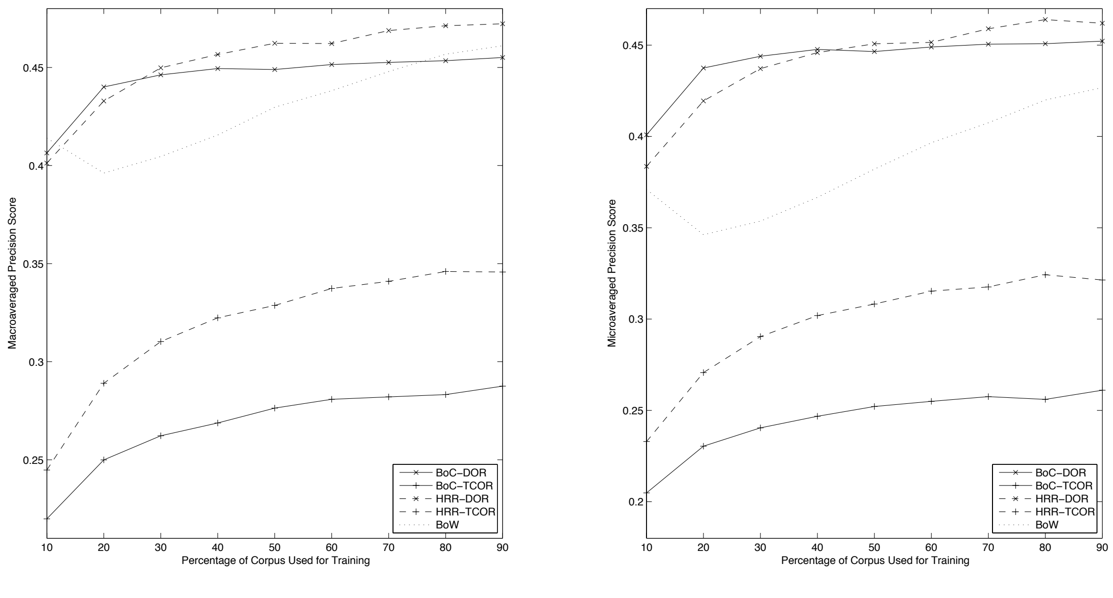
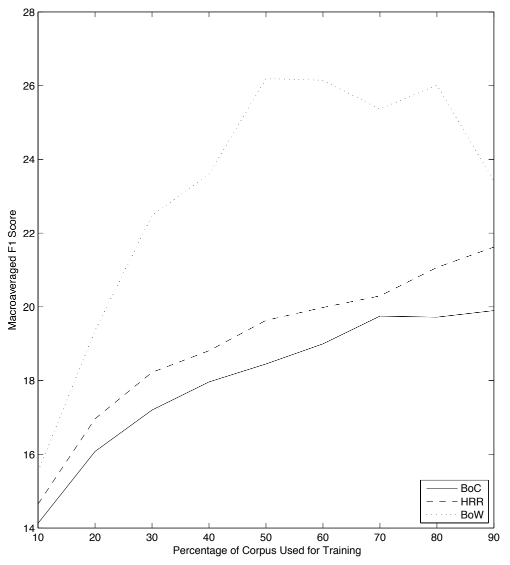


Figure 8.12: Macroaveraged and Microaveraged SVM Precision Scores of DOR and TCOR Generated BoC and HRR Representations and BoW Representations for the 20 Newsgroups Corpus.



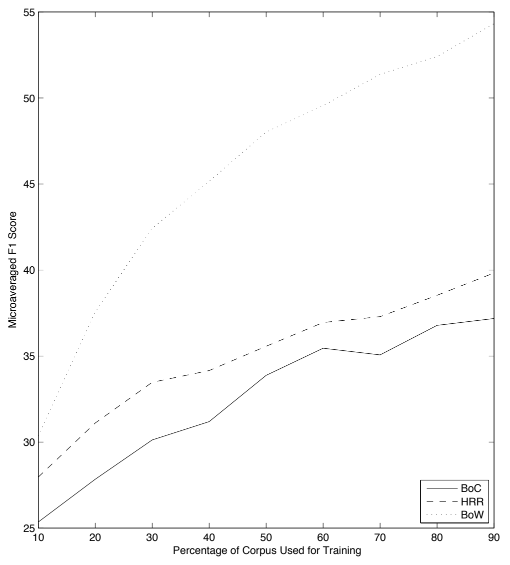


Figure 8.13: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of DOR Generated BoC and HRR Representations and BoW Representations for the Reuters-21578 Corpus.

and BoC representations produce more stable results between corpora.

Figure 8.15 shows the correlation graph between the 10-fold stratified cross-validation SVM $\mathcal{F}_1$ scores of the BoC-DOR and HRR-DOR text representations for each category in both the 20 Newsgroups and the Reuters-21578 corpus. In the 20 Newsgroups case, the graph shows that the HRR representations produce similar classification scores for some classes and significantly higher scores for other classes. This may be explained by noticing that the classes that the HRR representations outperform the BoC representations are the classes in the corpus that are highly related to other classes in the corpus (e.g.: `rec.sport.baseball` and `rec.sport.hockey` and `talk.politics.mideast` and `talk.politics.misc`). In the Reuters-21578 case, the graph shows that for the majority of the classes, the HRR representations produce better classification scores than the BoC representations. This is consistent with our previous suggestion that the HRR representations are better at disambiguating related classes, since the classes in the Reuters corpus all deal with finance related topics and hence use a similar vocabulary. In this case, the PoS information contained within the HRR representations has a greater positive effect in classification effectiveness over the BoC representations, which rely on semantics alone. Such results make sense given that the minimal structure we have encoded in the HRR representations is intended to help disambiguate the same word used in different ways. However, this hypothesis needs to be experimentally investigated in more detail.

Figure 8.16 shows the correlation graph between the 10-fold stratified cross-validation SVM $\mathcal{F}_1$ scores of the BoW and HRR-DOR text representations for each category in both the 20 Newsgroups and the Reuters-21578 corpus. For both

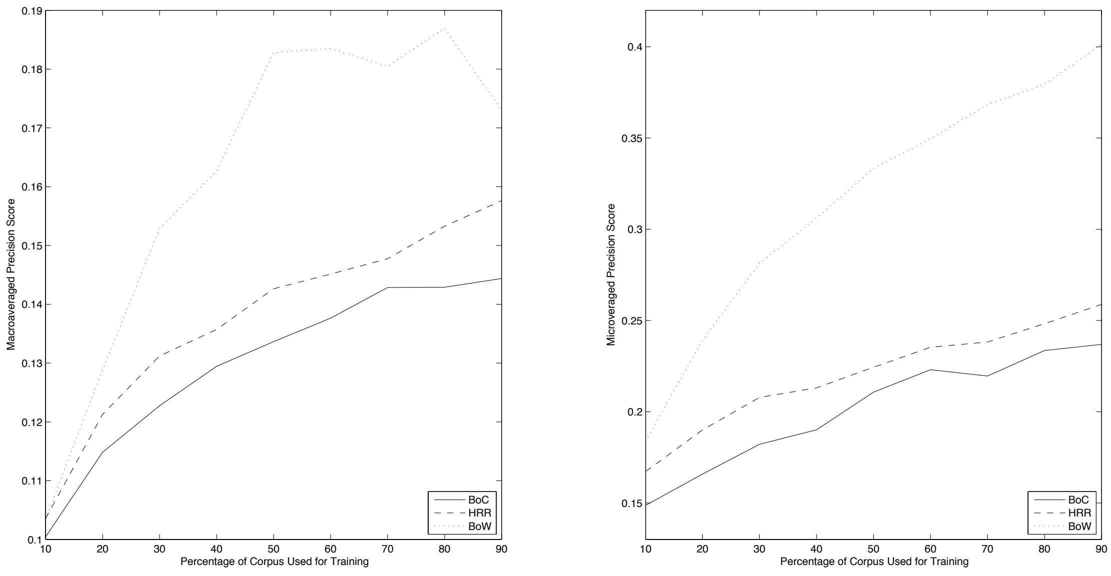


Figure 8.14: Macroaveraged and Microaveraged SVM Precision Scores of DOR Generated BoC and HRR Representations and BoW Representations for the Reuters-21578 Corpus.

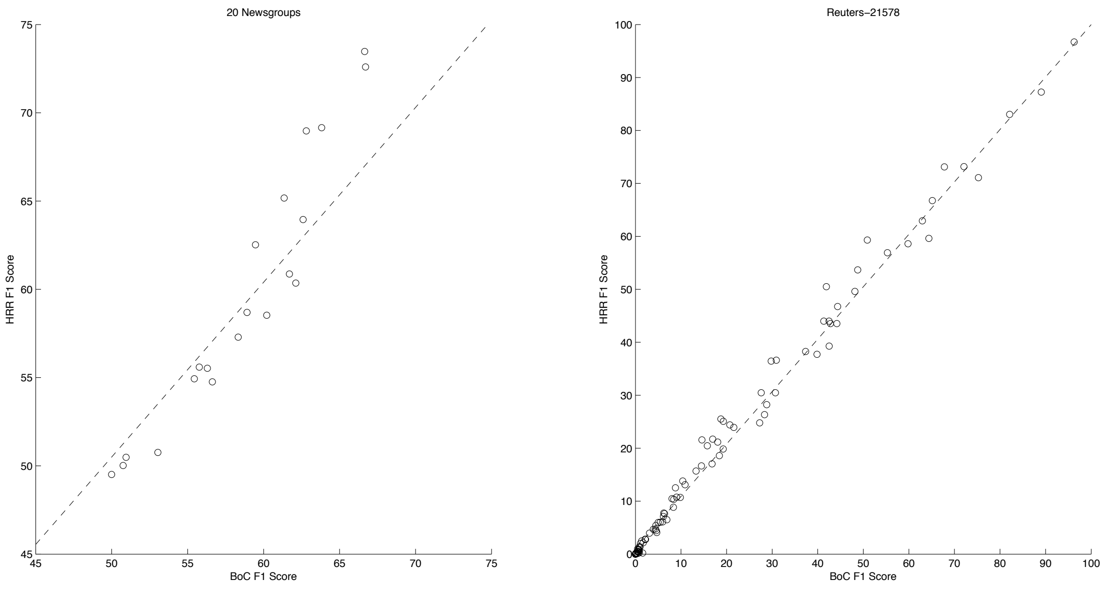
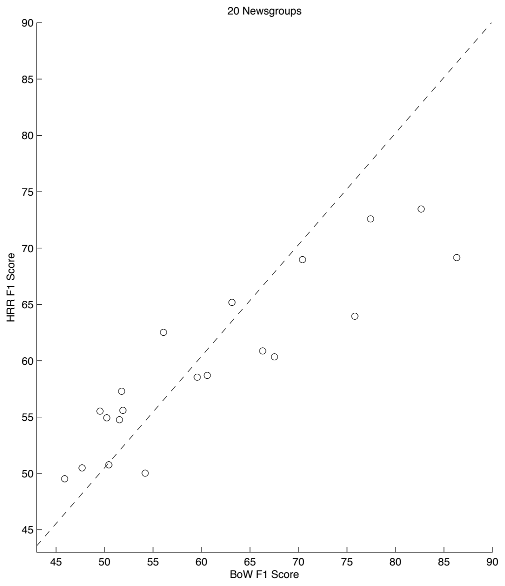


Figure 8.15: Correlation between SVM $\mathcal{F}_1$ Scores of BoC-DOR and HRR-DOR Text Representations for each Corpus Category in the 20 Newsgroups and Reuters-21578 Corpora with 90% of the Corpus Used for Training.



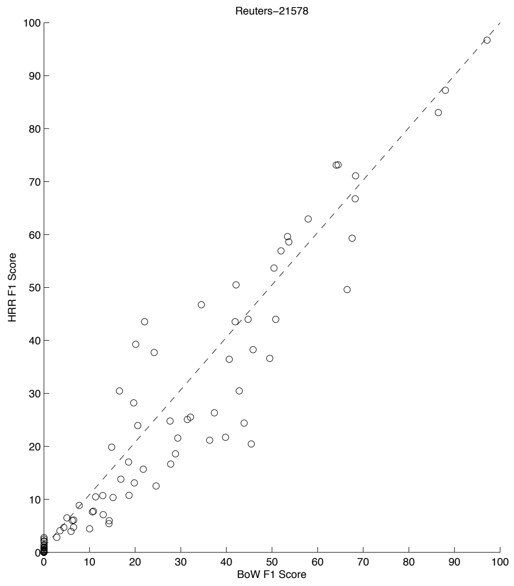


Figure 8.16: Correlation between SVM $\mathcal{F}_1$ Scores of BoW and HRR-DOR Text Representations for each Corpus Category in the 20 Newsgroups and Reuters-21578 Corpora with 90% of the Corpus Used for Training.

corpora, the majority of classes have higher classification scores under the BoW representations than the HRR representations. Also, unlike the correlation with the BoC representations, compared to the BoW representations there seems to be no pattern in which classes receive better $\mathcal{F}_1$ scores under the HRR representations.

Turning our attention to practical implications of these results, first we can conclude that DOR context definitions work better than TCOR context definitions for the BoC and HRR representations, regardless of the amount of training data used and regardless of the corpus. However, the decision of which representations to choose for our classification system is less clear. It seems that when training data is readily available, the BoW representations seem to perform the best. But, when data is limited for certain corpora, the BoW representations do not perform as well compared to the other representations, as illustrated by the results for the 20 Newsgroups corpus. For these corpora with limited data, the HRR-DOR representations perform better when a moderate amount of data is available for training, and the BoC-DOR representations perform better when there is a low amount of data available for training. One thing to keep in mind is the dimensionality of the representations. The BoC and HRR representations are of 1024 dimensions regardless of the corpus, while the BoW representations are respectively of 118 673 and 30 769 dimensions for the 20 Newsgroups and Reuters-21578 corpora. Since the methods for our classification system scale badly with representations of higher dimensions, the BoW representations take longer to classify than the BoC and HRR representations.

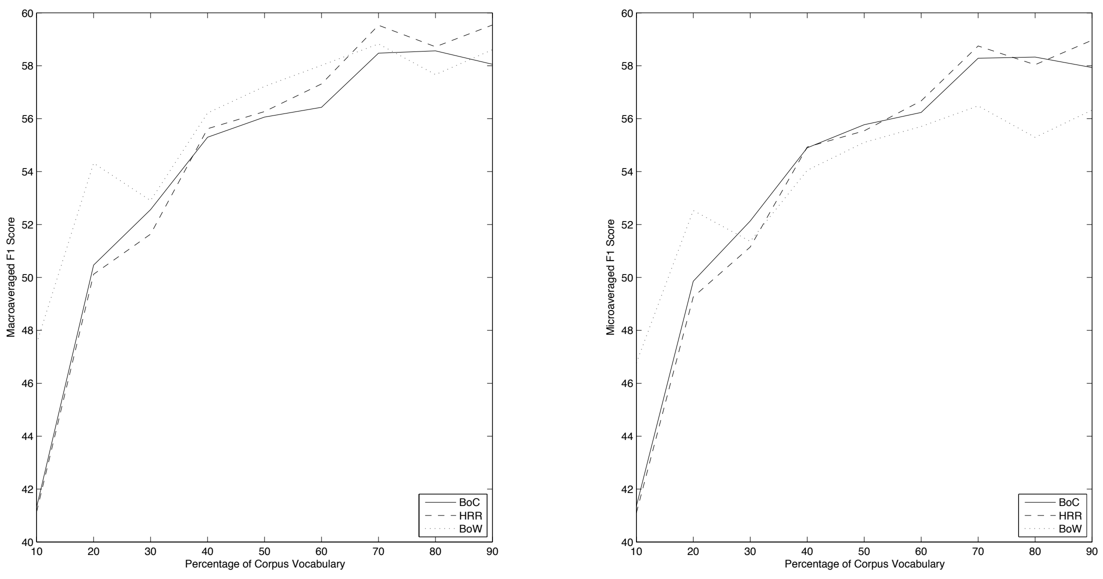


Figure 8.17: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of BoC, HRR, and BoW Representations for the 20 Newsgroups Corpus with a Limited Vocabulary with 90% of the Corpus Used for Training.

## 8.5 Limited Vocabulary Investigation

The $\mathcal{F}_1$ classification scores of the BoC, HRR and BoW representations for the 20 Newsgroups corpus under various percentages of the corpus vocabulary are illustrated in Figure 8.17. The interesting thing about these results is how the BoW representations are more greatly affected by the removal of percentages of the words in the corpus than the BoC and HRR representations. Although all the representations are negatively affected by this limiting of corpus words, the BoC and HRR representations show similar trends in results to the classification results without the limited vocabulary, as seen in Figure 8.10, while the BoW representation curves look the same but do not rise as high. Both the recall and precision scores show the same trends and have been omitted. These differences in the results are statistically significant to a 99% confidence interval using the $t$-test. The varational spikes seen in the results are due to the single and independent selection of the covubulary for a percentage level. Ecah percentage level had its vocabulary listing selected independently from the others and only selected once, even though the classification experiments were run multiple times using cross vlaidation. We suspect that if the voabulary lists for a percentage level were reselected with each classification run, these variational spikes would disappear.

Examining the $\mathcal{F}_1$ classification scores of the BoC, HRR and BoW representations for the Reuters-21578 corpus under various percentages of the corpus vocabulary, as illustrated in Figure 8.18, we see an even more interesting revelation. Again, the BoC and HRR representation results show similar trends to the non-limited vocabulary case, as illustrated in Figure 8.13, but the BoW representation

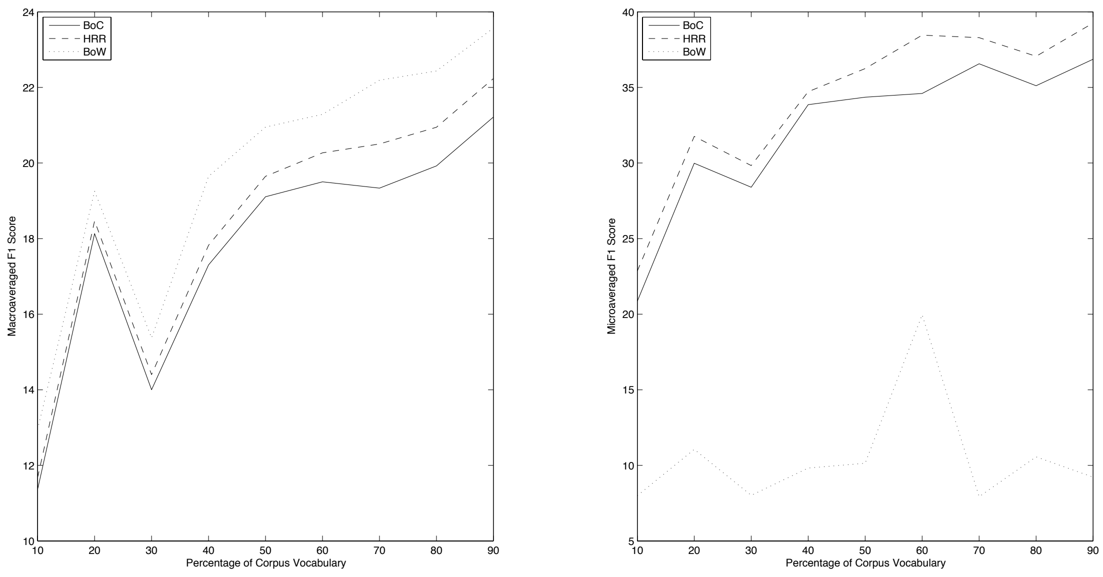


Figure 8.18: Macroaveraged and Microaveraged SVM $\mathcal{F}_1$ Scores of BoC, HRR, and BoW Representations for the Reuters-21578 Corpus with a Limited Vocabulary with 90% of the Corpus Used for Training.

results show significantly different results in the macroaverage and microaverage cases. Recall from Section 6.3 that macroaveraging combines all classes equally to obtain a single score, while microaveraging combines classes in proportion to their size to obtain a single score. What these results indicate for the BoW representation results is that the BoW representations do a good job at classifying the large number of small classes in the corpus, but fail to classify the small number of large classes where the majority of the documents reside in the corpus. This contrasts remarkably from the results of the full vocabulary case. Both the recall and precision scores show the same trends and have been omitted. These differences in the results are statistically significant to a 99.9% confidence interval using the *t*-test. Again, the variational spikes appear for the same reason as the last set of results, which we explained earlier.

Turning our attention to practical implications for classification system design, these results are a little harder to interpret. For one, there are very few reasons why a classification system designer would want to arbitrarily remove words throughout a corpus. However, this is a method for reducing the dimensionality of the BoW representations, as mentioned in the last section. From the results we see that the BoC and HRR representations remain more stable, while the BoW representation results are much more variable, especially in the Reuters-21578 case. This is not all that surprising, since the BoW representations rely on word surface forms as the only information for classification. Thus, the BoW representations will have less information to support classification when these surface forms are arbitrarily removed. Moreover, it is surprising that these effects are not more pronounced. Specifically looking at the BoC and HRR representations, it seems that the HRR representations have a small advantage in these tests over the BoC representations, especially when greater than 40% of the vocabulary is used for the 20 Newsgroups corpus and regardless of the amount of data used for the Reuters-21578 corpus.

# Chapter 9

# Conclusions

In this chapter, we give a brief summary of the insights reached from the research covered in this thesis. We first highlight what we consider are the most important positive and negative implications of our research. We conclude this section by providing some suggestions for future research.

## 9.1 Main Results

Using Holographic Reduced Representations (HRRs), we have created a novel document representation scheme that encodes both the structure and the semantics of natural language text documents. Our method successfully avoids the limitations of earlier attempts to encode structure by keeping the resulting dimensions of the document vectors constant and by ensuring that the structure is encoded across all the document features while preserving the text semantics.

A significant result of this thesis is that the circular convolution method of binding a document's parts-of-speech (PoS) information to its semantic model was found to be the best of all the methods investigated. What is also interesting to note, is that all the methods that we investigated of incorporating PoS information gave better $\mathcal{F}_1$ classification scores than the standard Bag-of-Concepts (BoC) semantic models, when a reasonable amount of training data is available. This shows that our initial hypothesis that PoS information can be useful for automated text classification was indeed justified. Our corollary hypothesis, that the method of including PoS data has an effect in classification scores, also appears to have been well-founded.

Another key result of this thesis is that the classification effectiveness of our HRR representations peaks at is most suitable at a moderate 1024 dimensions. This can be contrasted with the Bag-of-Words (BoW) representations, which use a dimensionality equal to the number of surface forms in the corpus, usually on the order of 10 000 to 100 000 dimensions. Decreasing document vector size has important consequences for real-world applications, in terms of both storage and manipulation.

Furthermore, our results show that generating our HRR representations using a limited PoS tag set based on the natural language PoS usage of the words gives better $\mathcal{F}_1$ classification scores than using the full Penn Treebank tag set. This is due to the design of our HRR representations, which assume that each unique tag is independent of one another in the tag set. In the limited PoS tag set, this assumption is valid since related tags have all been collapsed to a single tag. But for the Penn Treebank tag set, related PoS tags exist in the tag set and the useful information between these related tags is lost in the overall representation.

Turning our focus to context definitions for word-space models, DOR context definitions preform significantly better than TCOR context definitions for both the BoC and HRR representations in text classification tasks. This would support Sahlgren's intuition [43], that broader context definitions tend to capture what a word is *about* rather than what it *means*. More importantly, HRR-DOR representations yield better $\mathcal{F}_1$ classification scores than BoC-DOR representations when a moderate amount of data is available for training.

A disappointing result is that our HRR representations do not show a sustained classification advantage over BoW representations in a variety of classification scenarios. Our HRR representations do perform better than the BoW representations when the vocabulary of the corpus is artificially reduced and when training data is significantly limited for the 20 Newsgroups corpus, but these results were not as widespread as we would have hoped. BoW representations do perform better than our HRR representations on the Reuters-21578 corpus regardless of training data, which calls into questions the robustness of using HRR representations over BoW representations on corpora with limited training data. Moreover, the practical reasons for artificially limiting a corpus' vocabulary in a classification system are not well-supported. This result is counter-intuitive, since BoW representations have very little linguistic motivations unlike our HRR representations. Clearly, there is still more research that needs to be done to improve text representations in ways that are consistent with and motivated by linguistic theory.

## 9.2 Future Research

Our results suggest many areas of further research. Generating document vectors with PoS data seems to be beneficial for classifying text documents, especially in disambiguating classes with a certain degree of semantic overlap. This result is encouraging since the encoding scheme used to generate the HRR representations, binding PoS tag vectors with term context vectors, is a rather simplistic scheme. Since HRRs can bind information contained in greater than two constituent vectors, more complicated encoding schemes could be devised to further increase the classification results of the HRR representations. For example, it would be possible to bind more refined syntactic information, such as dependency relations, or to use HRRs to compress sequence information into a vector representation as shown by Jones and Mewhort [21]. Investigations into these more complicated representations based on HRRs should be performed to see if the classification scores of these representations significantly improves.

Furthermore, an investigation into different types of SVM kernels would be useful to determine which kernel provides the best SVM classification results for the HRR representations. We have only investigated classification results using the linear SVM kernel since that kernel was suggested to be superior for text classification [57]. However, this suggestion was made in the context of BoW representations. It is possible that the different SVM kernels, such as the polynomial and radial basis kernel, could provide better classification results for our HRR representations under the appropriate kernel parameters. Therefore, an investigation should be conducted to determine whether this is the case.

Also, looking at the classification results over more corpora would be ideal to see how the trends discovered in this thesis continue to hold. Although we have investigated our HRR representation's classification effectiveness on the 20 Newsgroups and Reuters-21578 corpora, these are considered relatively small corpora compared to real world data sets. Further experiments on larger corpora, like the Reuters V1 or the Enron Email Dataset, should be completed to see how our HRR representations perform when more and varied data is available.

We have only investigated the usefulness of our HRR text representations for the English language. It is possible that other languages will have linguistic properties that are better or worse suited to our HRR representation method than the other text representations currently in the literature. As a result, the experiments conducted in this thesis should be redone using non-English corpora to see if the results found in this thesis hold for other languages.

We have, also, only investigated the suitability of our HRR text representations in text classification situations. Although we find text classification an interesting problem in natural language processing, it is not the only problem where our representation method can be applied and tested. For instance, it is likely that our HRR text representations is well suited for author identification systems due to the theoretical ability of our representation to capture an author's writing style. As well, it would be interesting to see the effectiveness of our HRR representation methods in information retrieval systems. Experiments into these other applications of our HRR text representations should be completed to determine the general applicability of our representation method in natural language processing tasks.

Finally, an investigation into the structure of the HRR representation vector space would prove extremely useful to determine where the deficiencies are in the HRR representation method and what can be improved in the construction of the space to help improve the model. As noted recently by Karlgren [24], the standard BoC vector space model is not well understood in terms of structure. The same is true of our HRR vector space model, especially considering that we are adding more information to the standard BoC vector space using circular convolution. Continuing to explore the HRR representation space using some of the methods outlined by Karlgren, or by using artificially created corpora meant to test certain encoding properties of the space, will provide more insight into what kind of information is being stored within the vector space and how that information is stored. As well, investigating the similarity properties of the constituent vectors in the HRR representations, such as creating tag vectors with certain degrees of similarity, would be an interesting endeavour.

# References

[1] D. Achlioptas. Database-friendly random projections. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 274–281, New York, NY, USA, 2001. ACM Press.

[2] J. A. Anderson. A theory for the recognition of items from short memorized lists. *Psychological Review*, 80(6):417–438, 1973.

[3] E. Bingham and H. Mannila. Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining*, pages 245–250. ACM Press, 2001.

[4] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–565, December 1995.

[5] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In D. Yarowsky and K. Church, editors, *The Proceedings of the 3rd Workshop on Very Large Corpora*, pages 1–13, Cambridge, MA, USA, 1995.

[6] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there: Preference judgements for relevance. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *Advances in Information Retrieval: Proceedings of the 30th European Conference on IR Research*, volume 4956 of *Lecture Notes in Computer Science*, pages 16–27, Glasgow, UK, 2008. Springer.

[7] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, Cambridge, U.K., 2000.

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

[9] C. P. Dolan and P. Smolensky. Tensor product production system: a modular architecture and representation. *Connection Science*, 1(1):53–68, 1989.

[10] C. Eliasmith. Cognition with neurons: A large-scale, biologically realistic model of the Wason task. In B. Bara, L. Barsalou, and M. Bucciarelli, editors, *Proceedings of the XXVII Annual Conference of the Cognitive Science Society*, 2005.

[11] C. Eliasmith and P. Thagard. Integrating structure and meaning: A distributed model of analogical mapping,. *Cognitive Science*, 25(2):245–286, 2001.

[12] J. M. Fishbein and C. Eliasmith. Integrating structure and meaning: A new method for encoding structure for text classification. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *Advances in Information Retrieval: Proceedings of the 30th European Conference on IR Research*, volume 4956 of *Lecture Notes in Computer Science*, pages 514–521, Glasgow, UK, 2008. Springer.

[13] J. M. Fishbein and C. Eliasmith. Methods for augmenting semantic models with structural information for text classification. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *Advances in Information Retrieval: Proceedings of the 30th European Conference on IR Research*, volume 4956 of *Lecture Notes in Computer Science*, pages 575–579, Glasgow, UK, 2008. Springer.

[14] P. Frankl and H. Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory Series A*, 44(3):355–362, June 1987.

[15] R. Hecht-Nielsen. Context vectors: general purpose approximate meaning representations self-organized from raw data. In J. M. Zurada, R. J. Marks II, and C. J. Robinson, editors, *Computational Intelligence: Imitating Life*, pages 43–56. IEEE Press, 1994.

[16] G. E. Hinton. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–76, 1990.

[17] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kulwer Academic Publishers, Norwell, MA, USA, 2002.

[18] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, 2005.

[19] T. Joachims. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 217–226, 2006.

[20] W. B. Johnson and J. Lindenstrauss. Extensions to Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26, 1984.

[21] M. N. Jones and D. J. K. Mewhort. Representing Word Meaning and Order Information in a Composite Holographic Lexicon. *Psychological Review*, 114(1):1–37, 2007.

[22] P. Kanerva. *Sparse Distributed Memory.* The MIT Press, 1988.

[23] P. Kanerva. Binary spatter-coding of ordered k-tuples. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Artificial Neural Networks-ICANN Proceedings*, volume 1112 of *Lecture Notes in Computer Science*, pages 869–873, Berlin, 1996. Springer.

[24] J. Karlgren, A. Holst, and M. Sahlgren. Filaments of meaning in word space. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, editors, *Advances in Information Retrieval: Proceedings of the 30th European Conference on IR Research*, volume 4956 of *Lecture Notes in Computer Science*, pages 531–538, Glasgow, UK, March 2008. Springer.

[25] J. Karlgren and M. Sahlgren. From words to understanding. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 294–308. CSLI Publications, Stanford, CA, USA, 2001.

[26] A. Kehagias, V. Petridis, V.G. Kaburlasos, and P. Fragkou. A comparison of word- and sense-based text categorization using several classification algorithms. *Journal of Intelligent Information Systems*, 21(3):227–247, 2003.

[27] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1143, San Mateo, CA, USA, 1995.

[28] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[29] T. K. Landauer and S. T. Dumais. A solution to Plato's problem: the Latent Semantic Analysis theory for acquisition, induction and representation of knowledge. *Psychological Review*, 104(2), 1997.

[30] K. Lang. Newsreeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, 1995.

[31] A. Lavelli, F. Sebastiani, and R. Zanoli. Distributional term representations: an experimental comparison. In *CIKM '04: Proceedings of the thirteenth ACM conference on information and knowledge management*, pages 615–624, New York, NY, USA, 2004. ACM Press.

[32] D. D. Lewis. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318, February 1991.

[33] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behaviour Research Methods, Instrumentation and Computers*, 28(2):203–208, 1996.

[34] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, USA, 2002.

[35] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, June 1993.

[36] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[37] A. Moschitti and R. Basili. Complex linguistic features for text classification: a comprehensive study. In S. McDonald and J. Tait, editors, *Proceedings of ECIR-04, 26th European Conference on Information Retrieval Research*, pages 181–196, 2004.

[38] T. A. Plate. Holographic reduced representations: Convolution algebra for compositional distributed representations. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 30–35, San Mateo, CA, USA, 1991.

[39] T. A. Plate. *Holographic Reduced Representation: Distributed representation for cognitive structures*. CSLI Publications, 2003.

[40] D. A. Rachkovskij and E. M. Kussul. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13(2):411–452, 2001.

[41] F. Rosenblatt. The perceptron: A perceiving and recognizing automaton (project para). Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[42] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, USA, 2003.

[43] M. Sahlgren. Vector-based semantic analysis: Representing word meanings based on random labels. In *Proceedings of the ESSLLI 2001 Workshop on Semantic Knowledge Acquisition and Categorisation*, Helsinki, Finland, 2001.

[44] M. Sahlgren. An Introduction to Random Indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering*, 16August 2005.

[45] M. Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high dimensional vector spaces*. PhD thesis, Stockholm University, Stockholm, Sweden, 2006.

[46] M. Sahlgren and R. Cöster. 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*, pages 487–493, 2004.

[47] M. Sahlgren and J. Karlgren. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Journal of Natural Language Engineering*, 11(3):327–341, September 2005.

[48] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[49] F. Sebastiani. Machine learning in automated text classification. *ACM Computing Surveys*, 34(1):1–47, 2002.

[50] F. Sebastiani and F. Debole. Supervised term weighting for automated text categorization. In *Proceeding the 18th ACM Symposium on Applied Computing*, pages 784–788, Melbourne, FL, USA, 2003.

[51] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:471–496, 1990.

[52] D. Soergel. Indexing and retrieval performance: The logical evidence. *Journal of the American Society for Information Science*, 45(8):589–599, September 1994.

[53] V. Uren. An evaluation of text categorisation errors. In *Proceedings of the One-day Workshop on Evaluation of Information Management Systems*, pages 79–87, London, UK, September 2000.

[54] V. Vapnik. *Statistical Learning Theory.* Springer, New York, NY, USA, 1998.

[55] S. K. M. Wong, W. Ziarko, and P. C. N. Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25, 1985.

[56] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1-2):69–90, April 1999.

[57] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM Press.

[58] Y. Yang and J. O. Pederson. A comparative study on feature selection in text categorisation. In D. H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, Nashville, TN, USA, 1997.