

**Title line 1**

**Title line 2**

**A thesis submitted in partial fulfilment  
of the requirement for the degree of Doctor of Philosophy**

**Name M. Lastname**

**July 2011**

**Cardiff University  
School of Computer Science & Informatics**

**Declaration**

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

**Statement 1**

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

**Statement 2**

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

**Statement 3**

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Copyright © 2011 Name Lastname.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

A copy of this document in various transparent and opaque machine-readable formats and related software is available at <http://yourwebsite>.

**To People you care  
for their patience and support.**

# Abstract

We produce interpretable representations, and demonstrate their applicability in interpretable classifiers. Our approach is model-agnostic, given a similarity-based representation, we are able to produce a representation in terms of domain knowledge. We evaluate the interpretability of our representation and provide examples of interpretable classifiers with our representation.

# Acknowledgements

# Contents

<b>Abstract</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Contents</b>	<b>vi</b>
<b>List of Publications</b>	<b>xi</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Algorithms</b>	<b>xvi</b>
<b>List of Acronyms</b>	<b>xvii</b>
0.0.1 Definitions . . . . .	xvii
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Hypothesis . . . . .	3
1.3 Research Questions . . . . .	4
1.4 Contributions . . . . .	4

---

1.5	Thesis Structure . . . . .	6
1.6	Summary . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Introduction . . . . .	7
2.2	Text Data . . . . .	8
2.2.1	Text Domains . . . . .	9
2.2.2	Pre-processing Text Data . . . . .	10
2.3	Text Representations . . . . .	12
2.3.1	Bag-Of-Words . . . . .	12
2.4	Text Document Classification . . . . .	16
2.4.1	Multi-label problems . . . . .	17
2.4.2	Decision Trees . . . . .	17
2.4.3	Linear Support Vector Machines . . . . .	18
2.4.4	Neural Networks . . . . .	18
2.4.5	Overfitting . . . . .	19
2.4.6	Evaluation Metrics . . . . .	20
2.4.7	Low-Dimensional Vector Spaces . . . . .	21
2.4.8	Principal Component Analysis . . . . .	22
2.4.9	Multi-Dimensional Scaling . . . . .	23
2.4.10	Vector Space Representations Of Words . . . . .	23
2.4.11	Doc2Vec . . . . .	24
2.5	Interpretability . . . . .	24
2.5.1	Disentanglement and Conceptual Spaces . . . . .	25



---

2.6	Interpretable Representations . . . . .	26
2.6.1	Topic Models . . . . .	26
2.6.2	Generative Adversarial Network and Variational Autoencoders . . . . .	27
2.6.3	Sparse Representations . . . . .	28
2.7	Interpretable Representations . . . . .	29
2.8	Conclusions . . . . .	29
<b>3</b>	<b>Datasets and Semantic Spaces</b>	<b>31</b>
3.1	Introduction . . . . .	31
3.2	Datasets . . . . .	31
3.3	Technical Details . . . . .	34
3.4	Representations . . . . .	35
<b>4</b>	<b>Re-organizing Vector Spaces into Interpretable Representations</b>	<b>37</b>
4.1	Introduction . . . . .	37
4.2	The Use of Directions in Vector Spaces . . . . .	40
4.3	Background . . . . .	41
4.4	Clustering . . . . .	41
4.4.1	K-means . . . . .	41
4.4.2	Derrac's K-means Variation . . . . .	41
4.5	Method . . . . .	41
4.5.1	Obtaining Directions and Rankings From Words . . . . .	41
4.5.2	Filtering Word-Directions . . . . .	44
4.5.3	Clustering Features . . . . .	46
4.6	Experimental Method . . . . .	50

---

4.6.1	Document Embedding Models . . . . .	50
4.7	Qualitative Results . . . . .	50
4.7.1	The top-scoring directions for each domain . . . . .	50
4.7.2	Comparing Document Embedding Models . . . . .	51
4.7.3	Comparing Scoring Metrics . . . . .	53
4.8	Quantitative Results . . . . .	57
4.8.1	Evaluation Method . . . . .	57
4.8.2	Hyper-Parameters . . . . .	58
4.8.3	Summary of all Results . . . . .	62
4.8.4	Baseline Representations . . . . .	63
4.8.5	Word Directions . . . . .	65
4.8.6	Clustered Directions . . . . .	68
4.8.7	Conclusion . . . . .	71
<b>5</b>	<b>Directions in Neural Networks</b>	<b>75</b>
5.1	Introduction . . . . .	75
5.2	Background . . . . .	77
5.3	Method . . . . .	78
5.3.1	Inducing Rules from Auto-Encoder Entity Embeddings . . . . .	79
5.3.2	Feed-forward networks . . . . .	80
5.4	Investigating Feedforward Neural Networks . . . . .	81
5.4.1	Parameters . . . . .	81
5.4.2	Quantitative Results for Feedforward Network . . . . .	82
5.4.3	Qualitative Investigation of Feedforward Networks . . . . .	86

---

5.5	Qualitative Evaluation of Auto-encoders . . . . .	99
5.5.1	Software, Architecture and Settings . . . . .	99
5.5.2	Qualitative Evaluation of Induced Clusters . . . . .	101
5.5.3	Qualitative Evaluation of Induced Symbolic Rules . . . . .	101
5.6	Conclusion . . . . .	105
<b>6</b>	<b>Fine-tuning Vector Spaces to Improve Their Directions</b>	<b>106</b>
6.1	Introduction . . . . .	106
6.2	Fine-Tuning Vector Spaces And Their Associated Feature Directions . . . . .	109
6.2.1	Generating Target Rankings . . . . .	110
6.2.2	Generating Target Feature Values . . . . .	110
6.2.3	Fine-Tuning . . . . .	111
6.3	Quantitative Evaluation . . . . .	112
6.3.1	Results . . . . .	115
6.4	Conclusions . . . . .	116
<b>7</b>	<b>Appendix</b>	<b>117</b>
7.1	Chapter 3 . . . . .	117
7.1.1	Difference between Representations and Single Directions . . . . .	117
7.1.2	Class Names and Positive Occurrences . . . . .	119
	<b>GNU Free Documentation License</b>	<b>121</b>

# List of Publications

The work introduced in this thesis is based on the following publications.

- 
-

# List of Figures

- 4.1 An example of a hyper-plane in a toy domain of shapes. The hyper-plane for the word square is the dotted line. Green shapes are positive examples and red shapes are negative examples. Those closest to the hyper-plane are less square than those further away . . . . . 38
- 4.2 Another example of a hyper-plane in a toy domain of shapes. Here we show multiple directions, one for light and one for square. The hyper-plane for the word square is the dotted line. Green shapes are positive examples and red shapes are negative examples for the word square. . . . . 43
- 4.3 An example of a Decision Tree classifying if a movie is in the "Sports" genre. Each Decision Tree Node corresponds to a feature, and the threshold  $T$  is the required ranking of a document on that feature to traverse right down the tree instead of left. One interesting point to note is that the most important direction is used twice, the "coach, sports, team, sport, football" cluster and results in a majority of negative samples. Another point is that the nodes at depth three are more specific, sometimes overfitting (e.g. in the case of the "Virus" node, likely overfitting to a single movie about a virus) . . . . . 59
- 6.1 Toy example showing the effect of outliers in a two-dimensional embedding of geometric shapes. . . . . 107

# List of Tables

3.1	Text examples from three domains. For the movies and place-type domains, the original text was not available. . . . .	32
4.1	Example features from three different domains, where each cluster of words corresponds to a direction which movies are ranked on . . . . .	40
4.2	The top-scoring words for each domain. . . . .	52
4.3	Unique terms between document embedding models . . . . .	54
4.4	Different score types . . . . .	55
4.5	Comparing an MDS document embedding model to a D2V document embedding model for Newsgroups, where a D2V document embedding model performed best. . . . .	57
4.6	The F1-scores of trees limited to depth-one, two and three, for the best hyper-parameter variation of each representation . . . . .	64
4.7	Best results for the newsgroups for a variety of classifiers when using the document embeddings as input. . . . .	66
4.8	Best results for all domains for a variety of classifiers when using the document embeddings as input. . . . .	67
4.9	The best results (hyper-parameter tuned) for the classifiers for each task when the disentangled feature representations composed of rankings of entities on single-term directions. . . . .	69

4.10	All clustering size results for the newsgroups . . . . .	73
4.11	The best results when using the entities ranked on the directions output by the clustering method for the optimal number of clusters for each domain and task.	74
5.1	F1-scores for each embedding type and domain . . . . .	83
5.2	NDCG Top-Scoring directions . . . . .	88
5.3	Terms from three different document embeddings, the unsupervised embedding, the neural network that used a bag-of-words as input and the neural network that used the unsupervised vector space as input. Arranged by NDCG, from highest to lowest. . . . .	89
5.4	The largest differences in term scores between embeddings for the movies. The first two columns are the terms that were most reduced in score, while the third and fourth columns are the terms that increased most in score . . . . .	91
5.5	The top node of the Decision Tree for newsgroups . . . . .	93
5.6	The top node of the Decision Tree for Movies . . . . .	95
5.7	The placetypes clusters used at the top of the decision tree and the associated top-ranked entities . . . . .	98
5.8	Terms common for all embeddings and the associated ranking of entities on those term directions for each embedding. Arranged by NDCG-score in the unsupervised embedding. . . . .	100
5.9	A comparison between the first layers and the fourth layers of two different kinds of auto-encoders. . . . .	102
6.1	Comparing the highest ranking place-type objects in the original and fine-tuned space. . . . .	108
6.2	Results for 20 Newsgroups. . . . .	112
6.3	The results for Movie Reviews and Place-Types on depth-1, depth-3 and unbounded trees. . . . .	113

---

6.4	Results for IMDB Sentiment. . . . .	114
7.1	The difference between the representations being directly input to the low-depth decision trees and the word directions . . . . .	118
7.2	Positive Instance Counts for each Class . . . . .	120



## List of Algorithms

# List of Acronyms

**ML** Machine Learning

**NLP** Natural Language Processing

**NDCG** Normalized Discounted Cumulative Gain

## 0.0.1 Definitions

**Domain** Where the data was originally sourced from  $DOM^I MDB$ , e.g. IMDB movie reviews.

**Word** A string of alphanumeric characters that originated from text in the domain  $DOM_w$ , e.g. the  $w = "Horror"$  from a domain of IMDB movie reviews  $DOM^I MDB$ .

$w$

**Corpus of Documents** A unique group of words, e.g. a review from a domain of IMDB movie reviews  $DOM_I MDB$ .

$C_d w$

**Document** A document of words

$d_w$

**Vector Space** A representation composed of vectors.

$S_v$

**Semantic Space** A representation where spatial relationships between vectors correspond to semantic relationships.

$S_v$

**Word frequency** The frequency of a word  $w$  for its document  $D_w f$ .

$wf$

**Bag-Of-Words** a matrix BOW of documents  $BOW_D$  where each document is composed of unordered frequencies of words  $D = [wf_1, \dots, wf_n]$ . and Conceptual Space we obtain a representation of entities composed of properties. Then, we cover the additional methods we propose to improve this process.

$BOW_d$

**Bag-Of-Words PPMI**

**Feature** A feature is a distinct useful aspect of the domain, corresponding to a numerical value.

$R_f$

**Hyper-plane** The hyper-plane for a word

$H_w$

**Direction vector** The orthogonal direction to a hyper plane that separates a word in a vector space.

$D_w$

**Cluster label** A cluster of words that describe a property.

$C_w$

**Cluster direction** The averaged directions of all words in the label.

$D_C$

**Feature rankings** The rankings induced from a feature direction.

$R_D C$

---

# Chapter 1

## Introduction

### 1.1 Introduction

Applications that enable user-generated content e.g. Wikipedia, Social Media sites (Facebook, Twitter), Product and movie review sites (IMDB, Rotten Tomatoes, Amazon) and content-aggregation sites (Reddit, Tumblr) have resulted in widely available unstructured text data. This increased availability of text data has resulted in machine-learning models achieving state-of-the-art results on a variety of problems e.g. machine translation [?], question answering [?], or text classification like using text data to identify if social media posts or product reviews have a positive or negative sentiment about a company or product [?], to identify social media posts that are useful to crisis responders during crises [?], or to predict depression in social media users [?].

Historically in-order to achieve strong results in Natural Language Processing tasks like information retrieval, expert knowledge was used, e.g. a knowledge base [?]. However, as the volume of available text data has risen, machine-learning techniques that leverage that data have achieved the best results on the task [?], and eventually have become state-of-the-art in many fields<sup>1</sup>. However, these methods do not typically operate with just the raw unstructured text as input. This information is pre-processed into features that represent key domain knowledge. One simple method to obtain features is to use the frequencies of words. However, these statistics are not informative enough, leaving out information like word context.

One method to obtain features that represent complex domain knowledge while utilising a large amount of data is to induce a vector space. These vector spaces, or 'semantic spaces' represent

---

<sup>1</sup><https://github.com/sebastianruder/NLP-progress>

the semantic relationships between entities (words or documents) spatially. For example, word-vectors [?] [?] learned from a massive corpora like Wikipedia encode the meaning of words spatially by leveraging their context across millions of documents, resulting in e.g. spatial analogical relationships where  $\text{vec}(\text{man}) - \text{vec}(\text{king}) \approx \text{vec}(\text{woman}) - \text{vec}(\text{queen})$ . Similarly, vector spaces of text documents can be learned from unstructured text data and achieve strong results on many datasets with neural representation learning (e.g. doc2vec [?] or BERT [?]). However, the problem with these semantic spaces, especially those learned from neural networks, is that despite the representation encoding meaning spatially, features (i.e., the dimensions of the vector space) typically do not correspond to domain knowledge.

Feature representations and semantic vector spaces, are alternative and often competing representations of semantic relatedness [?]. These views are unified by conceptual spaces [?], which represent entities (e.g. in a conceptual space of fruit, entities are "orange" "apple" "watermelon") as points in a vector spaces where the dimensions correspond to primitive features of the domain (e.g. color, shape, taste in a conceptual space of fruit) and overlapping regions occur that correspond to properties of the domain (e.g. "tasty", "acidic", "bitter", "exotic").

Related to conceptual spaces is the idea of disentanglement in neural representation learning [?]. A disentangled representation is one where the 'factors of variation' have been spatially separated from each other, for example, separating style from content [?] [?], identifying key factors of interest in the medical domain [?], or e.g. in the 20 newsgroups domain where documents are separated into subjects like "Christianity" and "Motorcycles", obtaining features that correspond to these concepts [?]. Disentanglement has many benefits, such as enabling transformation of a factor e.g. sentiment while leaving the content intact [?], as well as potentially leading to better generalization, increasing the potential for transfer learning and resulting in more efficient learning without a supervised signal [?] [?].

The prominent advantage of disentangled representations is how they benefit learning, in particular by providing an inductive bias in machine-learning models. Disentangled representations are also linked to the objective of interpretability. As machine-learning has extended into real-world domains like medicine and policing, the legality and risk of implementing systems that do not have easily understood features (e.g. vector spaces) has resulted in concerns of safety (providing the wrong decision in a high-risk domain), fairness (using properties like the race of a person to e.g. deny a loan), and transparency (being able to know what the model is

doing and improve it) have not been accommodated. The EU have introduced a legal "Right to explanation", requiring that machine-learning models must be able to explain why they have made a decision about a person.

This thesis combines the ideas from conceptual spaces and disentangled representations. It follows work by Derrac [?] that followed the assumption that modern vector space methods act as conceptual spaces with regions of properties, and identified a method to induce features that correspond to properties of entities from the spatial relationships of vector space embeddings (e.g. in a domain of IMDB movie reviews a feature of "Comedy" would be a property of movies). These features are rankings of entities, derived from vector directions in the representation that go from entities that least have a property (e.g. movies that are the least "Funny") to those that have it the most.

Previously, disentanglement has required particular neural network architectures and learning methods, typically by learning a representation with a requirement that features must be independent from each other [?] [?]. Essentially, this thesis investigates the use of the method introduced by Derrac [?] as an unsupervised post-processing step to obtain a disentangled representation from existing vector space embeddings, by using the spatial structures of the vector space as features.

For this method, disentangled features are labelled, and e.g. in a domain of movie review documents where movies are represented by a concatenation of all their reviews, words like "Scary" or "Comedy" would be properties that movies are ranked on. However, this thesis does not attempt to validate the interpretability of these associated words or clusters of words. Rather, the focus is on validating that these features are indeed disentangled by testing them on tasks where dense but disentangled features (where the features must correspond to an important domain property and also encode a large amount of information) would perform well.

## 1.2 Hypothesis

Vector spaces of text documents encode semantic relationships spatially, e.g. in a domain where documents are amazon product reviews, a vector space that is successful at sentiment analysis will be organized such that documents that are negative (i.e. a one-star review) about the

product are distant from those that are positive (i.e. a five-star review), and there will be reviews inbetween (two, three or four star reviews). Vector spaces can be re-organized so that these semantic relationships are used as features, resulting in a disentangled feature representation. This disentangled feature representation will provide a useful inductive bias to classifiers and as it disentangles these properties, it can be used to qualitatively investigate the hidden layers of neural networks to gain valuable insights into how they represent documents.

## 1.3 Research Questions

**Question 1:** How can directions that rank documents on words in a vector space e.g. In a vector space constructed from the raw text of Amazon Product Reviews where documents are reviews, ranking documents on how "good" they are where the most negative review of the product is the lowest ranked and the most positive review is the highest rank, be used to achieve good disentangled representations of documents across a range of domains and from a variety of vector space embeddings?

**Question 2:** How can these directions that rank documents on words be used to gain qualitative insights into the characteristics of different neural networks?

**Question 3:** How can these directions that rank documents on words be improved in an unsupervised way?

## 1.4 Contributions

In Chapter 6, the method to re-organize vector space embeddings into disentangled feature representations introduced by Derrac [?] is subject to an extensive qualitative and quantitative analysis in five different domains and four different unsupervised methods for obtaining vector space embeddings from raw text data. The disentangled feature representations are validated quantitatively using Document Classification on key domain tasks, e.g. classifying the genres of movies based on the raw text data of their reviews. The classifier used is a low-depth Decision Tree, limited to a depth of one, two or three. This simple classifier is chosen as it means that predictive performance corresponds to disentanglement, as if the classifier can perform well

at a key domain task with e.g. a single feature in a one-depth decision tree, then that feature must have disentangled the key domain concept that the class represents e.g. when classifying if a movie belongs to the Horror genre there must be a feature that models if a movie is a Horror movie. Similarly, in a depth-2 limited tree the features must correspond to aspects of that class, e.g. when classifying if a movie belongs to the Horror genre there must be features that correspond to if the movie has properties like "Scary" or "Bloody". The method is found to be robust across the domains and variants of the method are introduced for better performance.

In Chapter ?? the method described in Chapter 6 is used to qualitatively investigate the hidden layers of neural networks, specifically feed-forward networks and auto-encoders. In particular, feed-forward networks are trained on key domain tasks, and the hidden layers of those trained models are re-organized into disentangled feature representations. These disentangled feature representations are quantitatively tested using depth-3 decision trees and the predictive performance of these decision trees is compared to the neural networks. It is found that in some domains, these trees match or even out-perform the neural networks that they came from. A qualitative investigation of the disentangled features from feed-forward networks is conducted and they are found to be meaningful. Finally, auto-encoders are used to obtain a sequence of increasingly small vector space embeddings with the understanding that each one will model more abstract concepts, and these layers are qualitatively investigated using directions. The ability to find relationships between these layers using the properties is investigated qualitatively by attempting to build a domain theory that goes from specific properties in the earlier layers to more abstract ones in the later layers. This work was published in NeSy'16, the Eleventh International Workshop on Neural-Symbolic Learning and Reasoning.

In Chapter ?? the conclusion is made from the results of the previous two chapters that the similarity centred objective used to build the vector space embedding, although resulting in meaningful directions can sometimes be counterproductive. Following this, a method is introduced to improve the directions in the vector space embedding at the expense of modelling similarity: First, Positive Pointwise Mutual Information (PPMI) scores for the words that label the interpretable features are obtained. Then, a target ranking for each feature is found by using isotonic regression to obtain values inbetween the PPMI scores and the rankings of the entities. This target ranking is used to train a single layer neural network with a non-linear activation function that attempts to match the rankings of entities to the target ranking. The intention



is not to achieve 100% accuracy, but instead rearrange the rankings so that similarity based information is de-prioritized over more meaningful directions. This results in a performance increase in some low-depth decision trees, and a qualitative investigation shows that the entity rankings become more specific and meaningful for the features. This work was published in The SIGNLL Conference on Computational Natural Language Learning (CoNLL) 2018.

## 1.5 Thesis Structure

- **Chapter ??** gives an overview of processing unstructured text data to obtain representations, some standard machine-learning classifiers, and provides a background on interpretable representations.
- **Chapter 3** introduces and explains the datasets used in this thesis, as well as giving an introduction to the hyper-parameters used for the machine-learning models in this thesis.
- **Chapter 6** quantitatively and qualitatively investigates how the method introduced by [?] can produce disentangled representations of unstructured text documents.
- **Chapter ??** qualitatively investigates the use and application of this method to neural networks, in particular feed-forward networks and auto-encoders.
- **Chapter ??** introduces the method to improve the interpretable feature representation by prioritizing these features over similarity information in the vector space.
- **Chapter ??** provides conclusions on the contributions of this thesis.

## 1.6 Summary

This thesis experimentally validates and improves on a methodology for obtaining disentangled feature representations for text documents. It is validated using document classification tasks, and disentangled features from neural networks are qualitatively analysed. The method is unsupervised, acts as a post-processing step on vector space representations, and disentangles important semantic relationships in the space.

---

## Chapter 2

# Background

## 2.1 Introduction

This Chapter begins by explaining the general process required to solving tasks with machine-learning starting from raw text data. The steps of this process are expanded on in the later sections.

In this thesis a representation must be obtained from raw text data that is effective for the task of document classification. Document classification is the task of distinguishing between entities in a domain, where entities are e.g. movies in a domain of movie reviews, people in a domain of twitter posts, or reviews in a domain of product reviews. As the task is to separate entities, after data collection the first step of solving the task is to separate the corpus of text into a document for each entity. Then, the text is pre-processed so that noise is removed, "noise" in this case referring to information in the text that is not useful when solving the domains document classification tasks. For example, metadata like the e-mail of a movie reviewer in movie review text, or unnessecary punctuation and grammar. It's important to remove this information at this stage as raw text data is easy to manipulate and the the result of any modifications can be clearly seen. If we tried to remove this kind-of information after obtaining a representation, it would be a much more complex process.

One popular and simple representation is the bag-of-words. The bag-of-words represents an entity as a vector where each element corresponds to a unique word in the corpus. The values of these elements are usually some statistic related to the words importance in the document e.g. word frequency where if a word occurs five times in a document the value is five. One disadvantage of this representation is that it doesn't retain the context of words, another is that it is sparse, as each vector has an element for every word in the corpus and only some of them will

have a frequency above zero. This means that to store it and process it in memory efficiently, specialized data structures and machine-learning techniques must be used. However, it has the advantage of being easy to understand for humans as each element of the vector representation for each entity corresponds to a word.

Ideally the number of dimensions would be reduced while retaining the information. One method of doing this would be hand-crafted feature selection, where words which are identified by experts as not meaningful are not included as an element. However, if it is done manually it would take a large amount of time and require expert knowledge, and if this is automated a lot of useful information can be lost to make the size of the data manageable. An alternative approach is to use the vector similarity between entities, e.g. the similarity between their frequency BOW vectors, to produce a low-dimensional vector space, where entities are encoded such that their semantic similarity matches their spatial similarity. In this space, the vectors that correspond to entities are just co-ordinates in a e.g. 2000 dimensional vector space. However, this results in vectors whose elements are no longer interpretable in the sense the bag-of-words is, instead information is stored as similarity relationships between entities in the vector space.

As mentioned in Section ?? the main focus of this thesis is in transforming vector spaces into interpretable representations while retaining their information. This Chapter introduces the process of obtaining bag-of-words and vector space representations from text data and using them to solve machine-learning problems, as well as giving an introduction to related work in the interpretability of machine learning representations and models. To outline the process, first as covered in Section 2.2.2 the data is preprocessed so that unnecessary information is removed. Then some basic representations are obtained in Section 2.3.1, followed by more complex vector space representations in Section 2.4.7. To complete a standard machine-learning pipeline, Section 2.4 covers different machine-learning methods to solve problems using these representations. Finally, interpretable representations and classifiers are covered in Section ?? to give context in the literature for the work in the next three Chapters.

## 2.2 Text Data

This thesis is focused on producing interpretable representations from text data, and solving specific problems in text domains. In this section, the basics of what the text data is, terminology

associated with it and how it is preprocessed is described.

### 2.2.1 Text Domains

"Text domain" refers to a subject area that is unique in its vocabulary and structure. One example is the newsgroups domain (See Section 3.2), which is composed of online news discussion groups from 1995. In this domain, there are subject areas, topics and posts. Each subject area has topics that users create, and each topic has posts that users respond with. Within each of these subject areas, specific jargon and a unique structure specific to that subject area and the overall domain has developed. Below an example post is provided from the newsgroups domain that contains unique jargon like "NOEMS", "EMM386" and a unique structure e.g. signing the post with the persons name and a personal tagline for contacting them.

Has anyone else experienced problems with windows hanging after the installation of DOS 6? I have narrowed the problem down to EMM386.

If if remove (or disable) EMM386, windows is ok. If EMM386 is active, with NOEMS, windows hangs. If I use AUTO with EMM386, the system hangs on bootup.

Dave.

—

---

David Clarke ...the well is deep...wish me well...

ac151@Freenet.carleton.ca David\_Clarke@mtsa.ubc.ca clarkec@sfu.ca

These particularities to the domain are what makes the distinction between domain-specific text data to general text data. A machine-learning model will develop a representation of how to solve the task dependent on this data. If the model is given general text data examples to learn from, then it will miss out on domain-specific quirks that can help it solve the task e.g. when learning to identify if a newsgroups post belongs to the subject of windows, if the general examples do not use jargon like "AUTO" and "EMM386" then this important information will not be used. However, if the examples given are over-specialized then the model may place

excess importance on domain-specific quirks that are not actually meaningful, e.g. if in the training examples of posts about windows most users signed off their text-posts with an email that includes ".ca", meaning they are from canada, then the model may identify all posts that include ".ca" emails as about windows despite this simply being a strange quirk in the data.

Although language is universal, the individualities of text domains make solving problems efficiently within those domains often depends on a domain-specific machine-learning pipeline where both the representation and the machine learning model that will solve the problem are catered towards that domain. For example, twitter posts are significantly shorter than news-groups posts, and rely more on modern expressions of ideas e.g. using a joke format that others on the platform have used. Being able to make-use of these domain-specific insights somehow in the process is extremely important.

In this thesis we aim to introduce methods that can be used to a variety of domains and be used with a variety of machine-learning models, without labour from domain experts. In particular, we look at solving domain-specific tasks without catering the representation or the model to the domain using expert knowledge. With this in mind, the following sections will be focused on a more general pipeline that does not delve into domain-specific techniques.

### 2.2.2 Pre-processing Text Data

Text documents in a domain usually reflect task that needs to be solved. In a domain-specific classification task, e.g. identifying the genre of a movie in a domain of movie reviews from its review text, the subject of the task are entities in the domain, in this case movies. A natural way to arrange the data is to create a document for each entity that contains all of its related data. For example, putting all the reviews for one movie in the same document. This is what is meant by a document-based task, where the corpus is arranged into documents that correspond to entities in the domain. In this thesis, we focus on these document-based domain specific tasks.

To obtain a good representation of a corpus, the text data must be processed so that it contains as little noise as possible. What exactly noisy data is depends on the representation and the task, but for this thesis it can be seen as parts of the text data that are not meaningful when distinguishing between types of entities in the domain. Noisy text data can have a knock-on effect on the representations that are built from it, resulting in a much worse representation. If the email of

a movie reviewer was retained in the review text, that will not be useful information for a task related to the movie. Additionally, you could also see a word starting with an uppercase or lowercase as noise, as it is not information that will benefit the representation.

Being able to automatically remove this noise is an essential step of building a representation and solving machine-learning problems. The first stage of obtaining a bag-of-words is building a vocabulary  $W_w$ , composed of unique words  $w \in W$  from the corpus. In this vocabulary, it is important that words which have the same meaning are not treated as different words e.g. if the word "Dog" was considered to be different to the word "dog." then the vocabulary would be too noisy. There are standard methods for removing noise in a dataset. We describe them in the following bullet-points:

- Convert all text to lower-case, e.g. "The man and The Dog" converted to "the man and the dog"
- Remove all punctuation including excess white-space, e.g. "the man, and the dog..." converted to "the man and the dog"
- Using a predefined list of "stop-words", listed in full in Table ??, remove words that are not useful, e.g. "the man and the dog" converted to "man dog"
- Remove infrequent words, e.g. "man dog, dgo, dog man" converted to "man dog, dog man".
- Domain-specific pre-processing to remove metadata, e.g. removing emails from the end of movie reviews.

In this work and the representations used in this work, the rules above are applied to the corpus beforehand. The methods are standardized so there should not be many interesting differences in the work, and it will also still be replicable. In terms of removing words that were not frequent enough, words that did not occur more than once are removed. Although these rules are not universal, they are a good basis for computational methods of representing text data that do not rely on word-context and grammar. In the next section, we cover some methods for text representation and explain their basic utilities.

## 2.3 Text Representations

Humans can have an intuitive understanding of the semantics that are present in unstructured text, but machines do not. Text documents like news articles, product reviews or social media posts cannot be classified without first being represented computationally. Representations  $r$  are composed of features  $r = (x_1, x_2, \dots, x_n)$ , where ideally each feature  $x$  is meaningful in the domain. For example, meaningful features when determining the value of a house would be the amount of bedrooms  $x_1$ , and the amount of toilets  $x_2$ . An example vector from these examples would be  $[5, 2]$  for a house with 6 bedrooms and 3 toilets.

In the next Chapter of the thesis, as well as in section ?? how to make a representation that both humans and machines can understand is discussed. However, this section focuses on representations that are useful to machines when used for machine-learning, rather than being interpretable. In particular this section covers preprocessing data 2.2.2, sparse bag-of-words representations ?? and obtaining vector spaces 2.4.7.

### 2.3.1 Bag-Of-Words

The bag-of-words is a simple representation that can scale to an extreme amount of data. Although when looking to achieve state-of-the-art results representations that are more complex or tailored to the domain are used, with enough examples even a basic representation can have enough information to clearly distinguish between types of entities in a domain for a task.

The bag-of-words (BOW) ignores word context, instead taking the words that occur in each document and assigning a value to them in a matrix, e.g. where each word in a document is assigned a value for its frequency in that document. For example, a short document of text like "there was a dog, and a man, and the man, and the dog" would be translated into word frequencies "there: 1, was: 1, a: 2, and: 3, the: 2, man: 2, dog: 2". This representation is simple, and ignores word context, grammar and punctuation but is highly effective when using machines to solve problems using a large amount of unstructured text documents. The bag-of-words is an important part of the work of this thesis, serving as the foundation of more complex and interpretable representations.

As mentioned in the previous section, unnessecary parts of the data that are not meaningful for

the task should be removed. The bag-of-words is a representation that comes with the following assumption: the context of words is an unnecessary part of the data to perform well on the task. How correct this assumption is depends on the task, but despite this view being overly-simplistic the application and use of the bag-of-words (BOW) is broad. There are multiple ways to represent words in the BOW format, but the most common is by the frequency of the words in a document.

The natural structure for this kind of representation is that of a matrix, where rows are documents and columns are words in the domain as defined by their vocabulary. Specifically, text documents in a domain  $d \in D$  have an associated vocabulary of unique words across all documents  $w \in W$ . The bag-of-words  $B_D$  is a matrix where each document is a row, and each column is a word, where the value of each word for a document is the word's frequency in that document  $d = (wf_1, wf_2, \dots, wf_n)$  where  $wf(d)$  is equal to the frequency of a word in a document and  $n$  is equal to the number of unique words in the vocabulary for all documents  $w \in W$ . In terms of the general structure given above, our representation  $r$  is the bag-of-words, and the features  $r = (x_1, x_2, \dots, x_n)$  are the word frequencies.

### **Term Frequency Inverse Document Frequency (TF-IDF)**

There are two main problems of using frequency is that words which are frequent in the domain are given a higher value than words which are used frequently only in a single document. First, longer documents result in overall higher values than shorter ones. So for example if a Amazon product review was very long and repeated the word "good" 15 times, but the word "bad" 1 time, then compared to a short review that only used the word "good" one time the first product review is fifteen times as good as the second one. When building representations that use vector similarity (e.g. where the bag-of-words vectors are compared in similarity to each other) these kind of value adjustments are very meaningful, as the documents need to be normalized relative to each other.

The second problem is that words that are frequent in many documents are given equal importance to those that are frequent only in some documents. However, we are concerned with what distinguishes documents from each other so giving equal importance for example, in the domain of movie-reviews, to the word "movie" does not accurately represent how important



it is for the meaning of the movie. Rather, we would be interested in terms that are frequent for only that movie review, as for example if the term "gore" was frequent in only five different movies out of 15,000 then it is clearly important for those movies.

The idea that words which are infrequent overall but frequent for some documents are important can be applied to a bag-of-words using the Term Frequency Inverse Document Frequency (TF-IDF) formulae. The first part of TF-IDF is Term Frequency  $TF_{d,w}$ , which is a normalization of frequency that solves the first problem of larger documents being treated as more important than shorter ones.

$$TF_{(w,d)} = \frac{wf(d)}{\sum_n wf_n(d)}$$

Where  $wf(d)$  is the number of occurrences of word  $w$  in document  $d$  and  $n$  is the number of words overall in the vocabulary. Note that frequency is still important, its just that it is not important how frequent it is relative to other documents. The next part of TF-IDF is Inverse Document Frequency, which is a measure that rewards terms that have a low Document Frequency.

$$IDF_w = \frac{d_n}{df(w)}$$

Where  $df(w)$  is the amount of documents the word  $w$  has occurred in and  $d_n$  is the amount of documents in the corpus. Note that while Term Frequency measures the frequency of a term in a document relative to that documents length, Document Frequency measures the overall occurrences of the term across all documents, relative to the number of documents. Essentially, it measures how rare that term is for a document, rather than how rare it is for a word. Finally, the TF-IDF is just the Term Frequency multiplied by the Inverse Document Frequency.

$$TF - IDF = TF \times IDF$$

### Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information (PMI) comes from probability theory and information theory, and is a metric that measures how *dependent* two variables are i.e. what is the difference between the chances of the variables occurring at the same time and the chances of them occurring independently. In this case, it can be used to measure how dependent a word is on a document. Obviously it is not possible to determine a precise probability that a word will occur, so in practice the frequency of the word is treated as an approximation of the chance it will occur. In application, we can understand that the word "the" is not independent from the document - it is a word that is just as likely to occur in one document than another because its occurrence is not dependent on the document. However, in a domain of movie reviews a word like "thrilling" would be more dependent on its associated text document, as it would only occur for movies which are thrilling. The pmi value for a word  $w$  in a document  $d$  is given by:

$$pmi(w, d) = \log \left( \frac{p_{wf(d)}}{p_{wf} \cdot p_d} \right)$$

where  $P_{wf(d)}$  is equal to the chance of the word occurring in the document assuming they are dependent on each other

$$P_{wf(d)} = \frac{wf(d)}{\sum_{wf} \sum_d wf(d)}$$

and  $wf(d)$  is the frequency of a word for a document. To calculate the chance that a word will occur, we simply take the chance the word will occur in any document (estimated by its summed frequency) over all frequencies, and for the document we take the chance that the document will occur (represented by the sum of the frequencies of all words that occur in it) over all frequencies:

$$P_{wf} = \frac{\sum_d wf(d)}{\sum_{wf} \sum_d wf(d)} \quad P_d = \frac{\sum_{wf} wf(d)}{\sum_{wf} \sum_d wf(d)}$$

As this value can sometimes be negative when words are less correlated than expected, we use Positive Pointwise Mutual Information (PPMI), as we are only interested in words which are positively correlated.

$$ppmi_{wf(d)} = \max(0, pmi)$$

The PPMI BOW is the representation used often in this thesis for a simple representation of meaning in the domain. It forms the basis of more complex representations and is also sufficient as a simple interpretable representation.

## 2.4 Text Document Classification

Problems that machine-learning can solve can be split into two distinct categories, supervised and unsupervised. Supervised problems have some data that is labelled, and some that is not labelled. The goal of a supervised task is to assign labels to the data that is not labelled, by learning with the data that is labelled. For example, classifying if a twitter post is positive or negative. Unsupervised problems do not have any labels, and instead try to solve a problem just from unlabelled data. An example of an unsupervised problem would be producing a representation from raw text data. Machine-learning models can be used to solve these problems.

Text document classification is a supervised task that can be used for example to identify if text posts like social media posts or product reviews, are positive or negative [?], identify social media posts that happen during crises and automatically categorize them to be useful to responders [?], or detect infections acquired while patients are in a hospital .

Representations are used to learn how to separate different kinds of entities in a domain. This is called a classification problem. A classification problem requires labels (or "classes")  $c \in C$ . Labels can be understood as categories in the domain, e.g. in the domain of sentiment analysis on movie reviews, labels could be "very good", "good", "average", "bad", "very bad". Given a set of possible labels documents  $D$  and document/label pairs assigned a binary truth value  $(d, c) = 0, 1$  find a function with a classifier  $FUNCT$  that assigns unlabelled documents  $d \in D$  predicted labels  $(d, c_p)$  approximates an unknown target function that can accurately label any

document. For example, in a domain of movie reviews labelled with if that review is positive or negative, find a function that can determine if unlabelled movie reviews are positive or negative. In this case we use classifier to refer to the method to obtain the function.

If the classifier performs well and can predict a variety of unlabelled documents, we can infer that the representation must represent the domain's knowledge sufficiently for the task. This is why classification tasks can measure how good a representation is, if they can perform on key domain tasks like predicting the genre of a movie based on its movie reviews then they clearly represent fundamental semantic information about movies. As an example, the bag-of-words can be considered a good representation if the frequencies of sentiment-related words, like "good", "bad", and "thrilling" would be good enough to achieve reasonable performance, as a machine-learning classifier could determine rules based on the frequency of these relevant words, e.g. "IF good > 30, and thrilling > 20, THEN positive sentiment". The tasks that are solved in this thesis are all classification tasks.

### 2.4.1 Multi-label problems

### 2.4.2 Decision Trees

Decision Trees are a model that result in a tree composed of nodes. Each node is associated with a feature from the representation, and an threshold value  $T$ . In the case of a bag-of-words, the nodes of this Decision Tree will correspond to unique words in the corpus vocabulary that are relevant to the task. If the bag-of-words measured raw frequency, then the threshold value would be checking how often that word occurs in a document. When the tree is processing a document, if the value given in the feature for that document is larger than the threshold  $T$ , then the tree is traversed along the left side, otherwise it traverses right side. Eventually the traversal reaches the bottom of the tree, called a leaf node, and the final decision made on the threshold of the leaf node is the classification of the document.

The tree can be viewed as a hierarchy of importance for the class, with the most important features for classification at the top and the less-important ones below. When viewing a decision tree spatially, we can see it as dividing the space into regions and sub-regions for the feature-values, with the top node of the decision tree diving the space the most.

Decision Trees has nodes that correspond to features, so if these features are simple and easy to understand then the tree is also interpretable. Generally, simple low-depth decision trees are a good baseline for an interpretable classifier.

### 2.4.3 Linear Support Vector Machines

Treating the entities as points in a vector space, where the dimensions of that space are the features, a linear support vector machine finds a hyper-plane that maximizes the margin between entities belonging to different classes. To classify new entities, they are placed in this space and labelled according to which side of the line they fall on.

### 2.4.4 Neural Networks

Neural networks are a model that can be used to solve both supervised and unsupervised problems. One-kind of network that solves supervised problems is the feedforward network. This network has sequential layers composed of nodes, where each node in one layer is connected to every node in the subsequent layer. There are three kinds of layers, the first is the input layer, which has the same amount of nodes as the input vector space has features. Then, there are hidden-layers, which vary in size, and finally an output layer that has a number of nodes  $n$  equal to the amount of classes. In the case of a binary classification problem, it would have one node.

Essentially, each node has an activation threshold which determines if the value will be propagated through the network, and each connection between a neuron has a weight which this value is multiplied by. The process of learning the network is tuning these parameters so that given an entity with an associated class label, the network is able to classify that entity by making the output node as close to the class as possible. Note, that this could mean that the output is a probability of the class occurring, and a simple threshold is applied to determine the binary value. The nodes of each layer have an activation function, which is a function used on values that are propagated from the node. These functions can be linear or non-linear.

The main benefit of neural networks is in its versatility. If the problem is more complex, then more nodes can be used. If the problem is simple, then less nodes can be used. Hidden layers can be viewed as vector spaces, and the result of learning is that the position of entities in that

hidden layer are transformed non-linearly such that they are better organized for the objective, for example splitting apart entities that belong to different classes. This is the relative power of neural networks over other approaches, an objective can be specified e.g. text classification, and then a network can be trained that obtains vector spaces organized according to that task with ease.

This benefit also has a down-side, as neural networks have so many parameters (e.g. the number of nodes, the activation function, the weight initialization) it can take a long time to find the combination of parameters that enable the network to organize entities efficiently for the associated problem. However, this lets them perform well in a variety of tasks.

### 2.4.5 Overfitting

If a machine-learning model is given training data, then what stops that model from learning a function that simply maps each example to the given class label? In the case of a neural network, this behaviour can be stopped by limiting the amount of neurons available in the hidden layer, forcing the network to generalize the representation into a lower-dimensional vector space. However, the problem of overfitting to the examples given rather than learning a way to solve the problem in a general way is a persistent one in machine-learning tasks. To give an example, we may expect that if we trained a machine-learning model on some data, we would be able to achieve strong results on that data given the machine-learning model. However, if new examples were introduced then the model would fail. For example, when learning with a bag-of-words the model may realize that each document was written by a different user, and that users name is recorded in the document text. A simple function would be to say:

IF user\_name\_1 is > 0, THEN class = 1.

However, this is not actually learning any domain knowledge, it is simply overfitting to noise.

To solve the problem of overfitting, the data for a supervised problem is usually split into three parts:

**Training data** The training data are the examples that the model learns from. It is used only when creating the model, and is not used after the model has finished learning.

**Test data** The examples that the model uses to check if the function learned is correct.

**Validation data** A decision tree may perform better if it is shallow and limited in depth rather than unlimited in depth, as it will not introduce nodes that are overly specific to the training data. Validation data is used for parameter tuning, e.g. when determining how much to limit the depth of a decision tree, how good the parameter is would be evaluated on how well the model performs on the validation set. The separation of validation data from test data is just to ensure that we are not overfitting the parameters on specific examples.

### 2.4.6 Evaluation Metrics

To evaluate a model, the difference between the real labels of documents and the predicted features of documents is compared. However, the value of the model is in its ability to predict the labels of documents that are unlabelled. Typically, this problem is solved by splitting the documents into a training set and a test set. The training set is used when learning the model, and the test set is used to verify the model is working correctly.

Here, we assume we are classifying a single binary class, where positive labels are 1 and negative labels are 0. The most simple way to evaluate a model is by its accuracy  $a$ , where  $t_n$  is the number of correct predictions, and  $P_n$  is the number of all predictions.

$$a = \frac{t_n}{P_n}$$

However, this can give a misleadingly high score if for example, the dataset is unbalanced with many more negative labels than positive ones, and the model predicts only negatives. An example of where this would be the case is when classifying out of all social media posts, which ones are important for emergency responders to investigate. Although there are very few positive instances of this class, identifying those is very important. In the case of a model predicting only negatives, the accuracy would be high as the number of correctly predicted negatives  $tn$  is high, but the model has not actually learned anything, which we can tell by looking at the number of correctly predicted positives  $tp$ . For a metric that can take this into account, we must consider the number of incorrectly predicted positives (negatives classified as positive)  $fp$  and the number of incorrectly predicted negatives  $fn$ .

In this situation, the metric we would want to optimize would be recall. Recall  $rec$  is the proportion of true positives  $tp$  identified correctly.

$$rec = \frac{tp}{tp + fn}$$

In the case of a model predicting only negatives, the *rec* would be zero. Recall is useful in these situations where we are interested in how many false negatives *fn* there are. However, if the model is instead prioritizing positive predictions too much rather than negative ones, we can use precision *pre*

$$pre = \frac{tp}{tp + fp}$$

F1 score is the harmonic mean of recall and precision, it is used to balance and measure the recall and precision at the same time where they are equally important.

$$F1 = 2 \cdot \frac{pre \cdot rec}{pre + rec}$$

### 2.4.7 Low-Dimensional Vector Spaces

The bag-of-words (BOW) based on frequency statistics has the benefit of being easy to understand on a granular level, as each feature is a distinctly labelled word. However, it is sparse which requires specialist data structures and algorithms to store and process it efficiently. Ideally, the information in a bag-of-words could be represented in a lower number of dimensions without losing information. Low-dimensional vector-spaces are one way that these sparse representations can be converted into low-dimensional dense representations.

Some neural network representation learning methods do not rely on the bag-of-words representation and are not designed just to reduce its dimensionality, they are instead learned explicitly such that they are able to integrate new kinds of information, e.g. contextual information, character-level information or information from other data sources. This shows the versatility of the low-dimensional vector space. It is able to encode complex information spatially that a simple representation like a bag-of-words would have difficult integrating.

Low-dimensional vector spaces generally work by taking the semantic information that is in the sparse representation, and encoding it spatially such that entities that are semantically similar are close together. This creates a representation that contains many complex relationships, but these dense vector space representations usually no longer have features which are meaningful to humans. This is a trade-off when going from a sparse representation to a dense representation, the features are no longer meaningful.



This can lead to unexpected disadvantages when classifying text with a simple interpretable classifier, e.g. a low-depth decision tree. In a bag-of-words, terms that are particularly important for classifying the task could be selected as important features at the top of the tree. However, in a low-dimensional vector space the information that is suitable for classification is not sufficiently separated into a distinct feature, rather it is encoded in the spatial relationships of the vector space. This means that features will not be able to be appropriately selected for the representation, and a deeper tree may be required to achieve strong performance.

The main focus of this thesis is in how to re-organize rich semantic relationships encoded spatially in any vector space such that they are used as semantic features. This is essentially producing a new representation that uses the same information as the vector space, but instead has features that are semantically meaningful similar to how a bag-of-words has individual features for each word. However, the features are not words but instead semantic relationships in the space that correspond to properties of the domain. In this case, "properties" refers to aspects of entities in the domain, for example, in a domain of movie reviews the "comedy" property describes how comedic movies are.

### 2.4.8 Principal Component Analysis

Principal Component Analysis (PCA) is a linear dimensionality reduction method that is non-parametric, meaning that the method does not vary according to some given parameters. Given features e.g. a bag-of-words, it produces a vector space of a specified size  $n$ , where dimensions are ordered by semantic importance.

Essentially, PCA works by linearly combining features in-order to create new features that can differentiate entities well and are uncorrelated with previous features. This results in a new low-dimensional representation that retains information and has distinct semantic features. However, as these features are a linear combination of the previous features, they are generally not interpretable [?].

### 2.4.9 Multi-Dimensional Scaling

Multi-Dimensional Scaling (MDS) is a non-parametric dimensionality reduction algorithm that can be metric or non-metric. Metric MDS is linear, while non-metric MDS is non-linear. In this work, non-metric MDS is used. In the same way as PCA, the size of the output space is specified. As input, MDS takes a dissimilarity matrix of entities, where both rows and columns are entities and the values are the dissimilarity between those entities.

From a bag-of-words, the way to construct this dissimilarity matrix is by finding the dissimilarity between bag-of-words features for each entity. The disadvantage of this is that it can be very large given many entities, which means it is difficult to fit into memory. The end-result of MDS is a representation where entities that are semantically similar according to the input matrix are spatially close to each other, and semantically different entities are spatially distant from each other.

### 2.4.10 Vector Space Representations Of Words

Word-vectors are a method that obtain a vector space representation for the words in a corpus, rather than the documents. Given some pre-processed raw text the method creates a vector representation for each word. The method is unsupervised, resulting in word vectors generally being used by learning them from a large corpus of unannotated text from a variety of domains, and then applying them in domain-specific tasks.

There are a multitude of ways to obtain word-vectors, like through matrix factorization [?]. However, most modern methods that are used today are distributional methods like GloVe [?] and Word2Vec [?]. These representations learn representations of words using the context of its surrounding words. Essentially, the meaning of each word is determined only by context. These representations have been extremely useful, and have shown semantic coherence, for example showing in the representation that it is possible to model relations between words, e.g. the vector operation "King" - "Man" = "Queen".

### 2.4.11 Doc2Vec

Doc2Vec [?] extends the Word2Vec neural network method of learning word-vectors using their context such that a document representation is learned in tandem. Essentially, as well as learning from the word's context, the words are also learned according to what documents they are in. The document representation is built in the same way as the word representation, gradually being informed by the word context and document context.

## 2.5 Interpretability

Going from a sparse but simple representation like bag-of-words to a dense and complex representation like Doc2Vec is a big leap in performance for a variety of tasks. However, the simple interpretability of the features is lost when using low-dimensional vector spaces. The work in this thesis is about how to re-organize any vector space such that an interpretable representation is obtained where the features are interpretable.

But what exactly is meant by "interpretable"? The definition of interpretability is as varied as the methods that claim it. In this work, we do not try to pin down the definition of interpretability, but instead appeal to a few provable ideas. The first is that we are interested in the interpretability of features, not of the application of the overall representation in some real-world domain e.g. the domain of medicine. When interpretability is viewed in the sense of application, it depends on the consumer of the information, and we are not interested in proving that the representation produced by our method is certainly applicable to different real-world situations or people.

Additionally, we are not interested in verifying with users if the features that are obtained are described well. The primary objective of the work is to obtain features that are semantic, and correspond to the relationships represented in the associated vector space. To verify that these features are semantic, we check how well they perform on key-domain tasks in a classifier where only a limited number of features can be used. If the classifier can perform well with a limited number of features on a key domain task this ensures that they are both independent and effectively represent important properties of entities in the domain.

Despite these features performing well at key-domain tasks, even when limited to using only

a single feature to classify entities, it is not automatically clear what they mean. In-order to help elucidate this, the features are labelled with a cluster of words  $w \in C$  which directly correspond to the semantic meaning. This is done automatically, and is qualitatively shown to be meaningful. Essentially, as the features obtained are representing some property in the domain, domain knowledge is required to understand what the cluster label is referring to. For example, the cluster *vhs, old, dvd* does not have an immediate clear meaning to someone who is not aware that these words are used in the reviews of old movies that are released to DVD and VHS rather than being in the cinema.

The end-result of this process is to obtain a representation where each feature is a semantic property in the domain, labelled with a cluster of words. The value associated with the feature for each entity corresponds to the degree that it "has" that feature, e.g. if a movie in the domain of movie reviews had a high value for a feature labelled with *Gore, Bloody, Horror* then we can rightly assume that the movie will contain a lot of blood. These semantic properties are derived directly from the spatial relationships in the representation, enabling us to use the versatility of information available in a variety of vector spaces to obtain interpretable representations that contain the same information. Although a low information loss is a by-product of this method, the main goal is not information loss, but just that the features obtained are useful in the domain.

### 2.5.1 Disentanglement and Conceptual Spaces

USED IN INTRO

The notion of disentanglement was popularized in the field of representation learning by [?], who introduced goals for good representations, with the primary goal of 'disentangling the factors of variation'. Originally, this meant that spatially the properties in the domain that most determine the differences between entities formed clusters of domain knowledge distant from each other. However, the idea of disentanglement has extended into producing interpretable features [?] where the aim of the representation is to find disentangled features that are factors of variation.

This is very similar to the goal of the work in this thesis, and follows the inspiration of the work that preceded this one [?]. Within the field of cognitive science, feature representations and semantic spaces both have a long tradition as alternative, and often competing representations of

semantic relatedness [?]. Conceptual spaces [?] to some extent unify these two opposing views, by representing objects as points in vector spaces, one for each facet (e.g. color, shape, taste in a conceptual space of fruit), such that the dimensions of each of these vector spaces correspond to primitive features.

The main appeal of conceptual spaces stems from the fact that they allow a wide range of cognitive and linguistic phenomena to be modelled in an elegant way. The idea of learning semantic spaces with accurate feature directions can be seen as a first step towards methods for learning conceptual space representations from data, and thus towards the use of more cognitively plausible representations of meaning in computer science. Our method also somewhat relates to the debates in cognitive science on the relationship between similarity and rule based processes [?], in the sense that it allows us to explicitly link similarity based categorization methods (e.g. an SVM classifier trained on semantic space representations) with rule based categorization methods (e.g. the decision trees that we will learn from the feature directions).

Fundamentally, both of these views seek to find the essential components that determine why all entities vary in the domain, and use them as features. In the case of text processing which we investigate in this work, the factors of variation found correspond to clusters of words that represent properties in the domain. The representation is considered disentangled if the features obtained are interpretable and predictive when used in key domain tasks.

## 2.6 Interpretable Representations

### 2.6.1 Topic Models

The interpretable representation that is obtained by this method is composed of salient features in the domain, in this case salient meaning that they are well-represented in the vector space, where each of these features is described using a cluster of natural language terms. This is somewhat similar to Topic models like Latent Dirichlet Allocation (LDA), which learns a representation of text documents as multinomial distributions over latent topics, where each of these topics corresponds to a multinomial distribution over words [?]. Topics tend to correspond to salient features, and are typically labelled with the most probable words according to the corresponding distribution.

Compared to topic models, vector space models have the advantage that they are versatile in how they can be learned, enabling e.g. structured knowledge from the domain, or different kinds of data like images to be taken into account. Some authors have also proposed hybrid models, which combine topic models and vector space models. For example, the Gaussian LDA model represents topics as multivariate Gaussian distributions over a word embedding [?]. Beyond document representation, topic models have also been used to improve word embedding models, by learning a different vector for each topic-word combination [?].

Compared to topic models, our work leverages clustering and similarity methods to obtain the feature labels, and is a post-processing step to re-organize vector spaces such that their features correspond to the semantics they represent spatially. This gives the methods in this work the advantage of broad applicability. However, there are extensions of LDA that have been proposed to incorporate additional information as well, e.g. aiming to avoid the need to manually specify the number of topics [?], modelling correlations between topics [?], or by incorporating meta-data such as authors or time stamps [?, ?]. Nonetheless, such techniques for extending LDA offer less flexibility than neural network models, e.g. for exploiting numerical attributes or visual features. For comparison, in our experiments the standard topic model algorithm Latent Dirichlet Allocation (LDA) is used as a baseline to compare to the new methodology that transforms standard Vector Space Model representations.

### 2.6.2 Generative Adversarial Network and Variational Autoencoders

Generative Adversarial Network (GAN) [?] are neural networks that learn representations using a discriminator and a generator, where the generator attempts to reproduce an entity and the discriminator attempts to determine if that produced entity matches reality or not. This results in a 'latent space' that represents fundamental knowledge in the domain that is used to produce new entities. However, the features in this latent space are generally not interpretable. However, GAN's have been extended to produce an interpretable disentangled latent space, in particular InfoGan has shown that it can obtain interpretable features in the latent space where each feature corresponds to a salient factor, e.g. in a task of identifying what digit is written in an image of a handwritten digit, there are features for each digit and an additional digit used for the style of writing [?]. GAN's have also been applied in text [?, ?] with some success, despite being noted

as 'particularly difficult to train' in the text domain [?] even with advancements in this direction [?].

The approaches found in GAN's and our work share the desire for a disentangled representation of features that are meaningful in the domain. However, the interpretability of these latent variables is determined qualitatively by examining how adjusting these features produce a variety of different samples [?]. Although it is clear that these features do have some meaning in the representation and can be useful for other tasks, they do not really follow our idea of interpretability in that they do not have automatic and natural labels, often needing expert knowledge to determine what they represent.

### 2.6.3 Sparse Representations

Methods to obtain sparse and interpretable word vectors have been achieved by either adapting a learning method to include sparsity constraints e.g. non-negative sparse embeddings adapting matrix factorization with sparsity constraints [?] or [?] adapting neural networks. Alternatively, some work follows a similar line to ours in that they post-process existing dense embeddings [?] [?]. In the former category, This approach has also been extended to sentences [?], and follows the idea that PCA and other dense representations are effective at compressing information into a small number of dimensions, but this results in semantically incoherent features. Instead, a larger representation with similar performance but more dimensions and high semantic coherency of its features is learned. This way, information that was compressed into a small amount of dimensions previously has been disentangled into a larger number of features. However, this can sometimes come with a minor loss of performance, particularly when using a lower number of dimensions. The features of these representations are labelled using the top  $n$  highest-scoring words on the feature. Sparse interpretable representations have also been derived from sentences [?].

There are document representations that use sparsity constraints to obtain interpretable sparse representations like sparse PCA learned using the l1-norm, [?] [?] or Sparse MDS [?]. Compared to sparse representations, the methods in this thesis do also attempt to post-process a dense representation similar to some word-vector methods in-order to disentangle them, but it does not aim to produce a sparse representation that may perform poorly with a small number

of features. Instead, the objective of the representation obtained in this thesis is to perform well with a small number of features, under the assumption that if we are able to identify key properties in the domain as features then we should only need a small number of features to perform well at key domain tasks like text classification.

One method that does not produce a sparse representation but still learns an interpretable representation of word vectors is [?] which uses an external lexical resource to define concepts that will correspond to features in the representation before training. This differs from our work in that we do not use any external resources apart from a bag-of-words from the domain to determine the features, rather they are determined by what the vector space representation itself prioritizes, as the features are derived directly from the semantic relationships that are spatially encoded in the representation.

## 2.7 Interpretable Representations

## 2.8 Conclusions

The most commonly used representations for text classification are bag-of-words representations, topic models, and vector space models. Bag-of-words (BOW) representations are simple and meaningful, achieving strong results despite not being complex. BOW representations are also interpretable in principle, but because the considered vocabularies typically contain tens (or hundreds) of thousands of words, the resulting learned models are nonetheless difficult to inspect and understand. Further, the sparsity and size of this representation limits its applications. Topic models and vector space models are two alternative approaches for generating low-dimensional document representations, with the usual advantage of topic models over vector-space models being that their features are interpretable, as the features are labelled with a group of words. However, vector space models are used on a larger variety of tasks as they are very versatile, and can achieve state-of-the-art results.

Interpretability in this thesis is defined as achieving a disentangled representation where each feature is associated with a group of words that describe its meaning. GAN's seem promising in that they can achieve a disentangled representation, but they are difficult to train on text data



and lack automatic labelling techniques. Methods to obtain sparse interpretable representations in word-vectors are similar to this work in that they post-process a dense representation, but these methods are limited to word vectors and suffer in performance with low-dimensionality, which we identify as a desirable property of our representation. To the authors knowledge, there is not existing work on obtaining an interpretable document representation from a dense representation that does not utilize sparsity constraints.

This thesis continues as follows: given the background in this chapter, the datasets that will be used in text classification tasks and to produce the dense and interpretable representations are introduced. Then, the method to re-organize dense vector spaces into interpretable representations is deeply experimented on and quantitatively and qualitatively validated. Following this, the dense vector space representations of neural networks are investigated, with the intention to better understand these models with unexpected results. Finally, a method accepted into CONLL 2018 to improve both the semantic coherence and performance of these interpretable representations is introduced and quantitatively and qualitatively validated.

(Why not compare lol)

# Datasets and Semantic Spaces

## 3.1 Introduction

For the experiments in this thesis, five different domains are used, each with their own particular vocabulary and meaning of words in their vocabulary. This Chapter begins with a section to give insight into the datasets with explanations of each domain, accompanying examples, and their classes. This is followed by technical descriptions of preprocessing methods for the datasets. Finally, we introduce the bag-of-words and semantic space representations built from these preprocessed datasets that will be used in the remainder of the thesis.

## 3.2 Datasets

First, we go through the history and class names of the datasets to give context, and provide examples of unprocessed text from three domains in Table 3.1.

**IMDB Sentiment** Where documents are exclusively highly polar IMDB movie reviews, either rated  $\leq 4$  out of 10 or  $\geq 7$  out of 10. Reviews were collected such that it was limited to include at most 30 reviews from any movie in the collection, as some movies contained many more reviews than others. The corpus is split half and half between positive and negative reviews, with the task being to identify the sentiment of the review.

**20 Newsgroups**<sup>1</sup> Originating from online news discussion groups from 1995 called newsgroups, where group email-type discussions are made by users about particular topics within 20 different groups. In this dataset, each document is composed of a topic, where user posts are concatenated

---

<sup>1</sup><http://qwone.com/~jason/20Newsgroups/>

Data Type	Unprocessed	Processed
Newsgroups	morgan and guzman will have era's 1 run higher than last year, and the cubs will be idiots and not pitch harkey as much as hibbard. castillo won't be good (i think he's a stud pitcher)	morgan guzman eras run higher last year cubs idiots pitch harkey much hibbard castillo wont good think hes stud pitcher
Sentiment	All the world's a stage and its people actors in it--or something like that. Who the hell said that theatre stopped at the orchestra pit--or even at the theatre door? Why is not the audience participants in the theatrical experience, including the story itself? This film was a grand experiment that said: "Hey! the story is you and it needs more than your attention, it needs your active participation". "Sometimes we bring the story to you, sometimes you have to go to the story." Alas no one listened, but that does not mean it should not have been said."	worlds stage people actors something like hell said theatre stopped orchestra pit even theatre door audience participants theatrical experience including story film grand experiment said hey story needs attention needs active participation sometimes bring story sometimes go story alas one listened mean said
Reuters	U.K. MONEY MARKET SHORTAGE FORECAST REVISED DOWN The Bank of England said it had revised its forecast of the shortage in the money market down to 450 mln stg before taking account of its morning operations. At noon the bank had estimated the shortfall at 500 mln stg.	uk money market shortage forecast revised bank england said revised forecast shortage money market 450 mln stg taking account morning operations noon bank estimated shortfall 500 mln stg

**Table 3.1: Text examples from three domains. For the movies and place-type domains, the original text was not available..**

together. The groups that topics are categorized by are Atheism, Computer Graphics, Microsoft Windows, IBM PC Hardware, Mac Hardware, X-Window (GUI Software), Automobiles, Motorcycles, Baseball, Hockey, Cryptography, Electronics, Medicine, Space, Christianity, Guns, The Middle East, General Politics and General Religion, which also act as the classes for this dataset when being evaluated. Generally, it can be quite easy to identify if a document belongs to a particular group if it uses a keyword unique to that group, e.g. the word "chastity" will almost always mean that the document belongs to the "Christianity" class.

**Reuters-21578, Distribution 1.0** Text from the Reuters financial news service in 1987, composed of a headline and body text. The classes were chosen with assistance from personnel at reuters<sup>2</sup>, meaning that they can contain jargon. For that reason, explanations are provided with the original names in brackets. The classes are Trade, Grain, Natural Gas (nat-gas), Crude Oil (crude), Sugar, Corn, Vegetable Oil (veg-oil), Ship, Coffee, Wheat, Gold, Acquisitions (acq), Interest, Money/Foreign Exchange (money-fx), Soybean, Oilseed, Earnings and Earnings Forecasts (earn), BOP, Gross National Product (gnp), Dollar (dlr) and Money-Supply.

<sup>2</sup>For more detail on the history of the dataset: <https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection>

**Placetypes** Taken from work by Derrac [?]. Originating from the photo-sharing website flickr, where photos are tagged (i.e. words describing the photos like "sepia" or "mountain") by users. 22,816,139 photos were considered, and tags that occurred in place-type taxonomies (GeoNames, a taxonomy of man-made and natural features, Foursquare a mostly flat taxonomy of urban man-made places like bars and shops, and the site category for the common-sense knowledge base taxonomy OpenCYC) with more than 1,000 occurrences were chosen as documents. Each document, named after a flickr tag, is composed of all flickr tags where that tag occurred. There are three tasks, generated from the three different place type taxonomies. The Foursquare taxonomy, classifying the 9 top-level categories from Foursquare in September 2013, Arts and Entertainment, College and University, Food, Professional and Other Places, Nightlife Spot, Parks And Outdoors, Shops and Service, Travel and Transport and Residence. the GeoNames taxonomy limited to 7 classes, Stream/Lake, Parks/Area, Road/Railroad, Spot/Building/Farm, Mountain/Hill/Rock, Undersea, and Forest/Heath, and the OpenCYC Taxonomy, which we limited to 25 classes, Aqueduct, Border, Building, Dam, Facility, Foreground, Historical Site, Holy Site, Landmark, Medical Facility, Medical School, Military Place, Monsoon Forest, National Monument, Outdoor Location, Rock Formation, and Room. Naturally as these tasks were derived from taxonomies they are multi-label.

**Movies** Taken from work by Derrac [?]. The top 50,000 most voted-on movies were chosen for this dataset initially, and reviews were collected from four different sources (Rotten Tomatoes, IMDB, SNAP project's Amazon Reviews <sup>3</sup> and the IMDB sentiment dataset. Then, the top 15,000 movies with the highest number of words were chosen as documents, where each document is composed of all of that movies reviews concatenated together. Three tasks are used to evaluate this dataset: 23 movie genres, specifically Action, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film-Noir, History, Horror, Music, Musical, Mystery, Romance, Sci-Fi, Short, Sport, Thriller, War, Western. 100 of the most common IMDB plot keywords (See Appendix ??) and Age Ratings from the UK and US, USA-G, UK-12-12A, UK-15, UK-18, UK-PG, USA-PG-PG13, USA-R.

<sup>3</sup><https://snap.stanford.edu/data/web-Amazon.html>.

### 3.3 Technical Details

In this section, we describe the vocabulary and document sizes for each domain. Each domain is preprocessed such that it is converted to lower-case, non-alphanumeric characters are removed and whitespace is stripped such that words are separated by a single space. Words are removed from a standard list of English stop-words from the NLTK library [?] and we filter out terms that do not occur in at least two documents, with an additional limit to the maximum number of words in a vocabulary set to 100,000.

**IMDB Sentiment**<sup>4</sup> When the original corpus was produced, the 50 most frequent terms were removed. It contains 50,000 documents with a vocabulary size of 78,588. After removing terms that did not occur in at least two documents, the vocabulary size was reduced to 55384. the number of positive instances in the classes is 25,000.

**20 Newsgroups**<sup>5</sup> Obtained from scikit-learn. <sup>6</sup> Originally containing 18,846 documents, in this work it is preprocessed using sklearn to remove headers, footers and quotes. Then, empty and duplicate documents are removed, resulting in 18302 documents. The vocabulary size (unique words) is 141,321. The data is not shuffled. After filtering out terms that did not occur in at least two documents, we end up with a vocabulary of size 51,064. This is a larger change than the sentiment dataset, despite beginnging with a larger vocabulary, likely because newsgroups contains many terms that were not relevant to a majority of the documents, instead being particular to their groups. The number of positive instances averaged across all classes is 942, around 5%.

**Reuters-21578, Distribution 1.0** Obtained from NLTK<sup>7</sup> originally containing 10788 documents. After removing empty and duplicate documents the result is 10655 documents. Originally contained 90 classes, but as they were extremely unbalanced all classes that did not have at least 100 positive instances were removed, resulting in 21 classes. The original vocabulary size is 51,001 and all words that did not occur in at least two documents were removed, resulting in a vocabulary size of 22,542. The number of positive instances averaged across all classes is 541, around 5%.

---

<sup>4</sup>Obtained by: <https://keras.io/datasets/>, Originally from <https://ai.stanford.edu/amaas/data/sentiment/> [?]

<sup>5</sup><http://qwone.com/~jason/20Newsgroups/>

<sup>6</sup>[https://scikit-learn.org/0.19/modules/generated/sklearn.datasets.fetch\\_20newsgroups.html#sklearn.datasets.fetch\\_20newsgroups](https://scikit-learn.org/0.19/modules/generated/sklearn.datasets.fetch_20newsgroups.html#sklearn.datasets.fetch_20newsgroups)

<sup>7</sup><https://www.nltk.org/book/ch02.html>

**Placetypes** It originally has a vocabulary size of 746,527 and 1383 documents. This is a very large vocabulary size to document ratio. The end vocabulary for this space was of size 100,000 due to the hard limit. This is roughly equivalent to removing all documents that would not be in at least 6 documents. As most classes in this domain are extremely sparse (less than 100 positive instances) no classes are deleted. As 8 of these remaining classes had a low number of positive occurrences, OpenCYC classes are removed that do not have positive instances for at least 30 documents, leaving us with 17. For the Geonames taxonomy, the same rule resulted in only 7 of 9 categories being used.

**Movies** Another large dataset with a vocabulary size of 551,080 and a document size of 15,000. However, after investigating the data made available by the authors, it was found that there were a number of duplicate documents. After removing these duplicate documents, there are 13978 documents. In the same way as the place-types, the vocabulary hit the hard limit of size 100,000.

## 3.4 Representations

For the bag-of-words representation used as a baseline, terms are additionally filtered out that do not occur in at least 0.001% of documents, as to scale with the amount of documents in each domain. From this filtered vocabulary, a bag-of-words is obtained by creating a matrix of documents and words, with the values of that matrix corresponding to how frequent each word was for each document. However, as frequency bag-of-words are not able to distinguish between frequent terms (e.g. "the") and important terms, words are weighed such that words which occur frequently in a small amount of documents are given a higher value than those that occur frequently in a large amount of documents. To do this, Positive Pointwise Mutual Information (PPMI) scores are used, following success in similar work by [?]. See section ?? for more detail.

For the work in the following chapters, we wanted a variety of different Vector Space Models. Below the choices for the Vector Space Models that are formally described in Section ?? are explained:

**Multi-Dimensional Scaling (MDS) (See Section 2.4.9):** Multi-Dimensional Scaling (MDS) is used for comparison, as it was the only space used in the work that introduced this method [?].

In this case, the input is a matrix of dissimilarity values between the PPMI vectors of documents of size  $n \times n$ , where  $n$  is the number of documents.

**Principal Component Analysis (PCA) (See Section 2.4.8):** We use PCA as a linear transformation of the PPMI weighted BoW vectors, as it is a standard dimensionality reduction technique used historically and prevalently today to serve as a baseline reference.

**Doc2Vec (D2V) (See Section 2.4.11):** Doc2Vec is inspired by the Skipgram model [?]. It is distributional in the sense that the context of words and documents is used during its learning process. It is used here as it is of a recent class of neural embedding models, which has been reported in the literature to perform well in document classification tasks. For the Doc2Vec space, the following hyper-parameters are tuned:

- The *window size*(5, 10, 15) referring to the window of the words that are used as context during training
- The *mincount*(1, 5, 10) referring to the minimum frequency of words
- The *epochs*(50, 100, 200) of the network for each size space.

**Average Word Vectors (AWV):** Finally, we also learn a document embedding by averaging word vectors, using a pre-trained GloVe word embeddings (See Section 2.4.10) that was trained on the Wikipedia 2014 + Gigaword 5 corpus<sup>8</sup>. While simply averaging word vectors may seem naive, this was found to be a competitive approach for unsupervised representations in several applications [?]. We simply average the vector representations of the words that appear at least twice in the BoW representation. Strangely, we found that this performed better than weighing the words on frequency or PPMI.

To determine the best doc2vec model for each task in each domain, and to investigate the quality of these representations, a linear SVM is used that takes the document representations as input. This SVM is also hyper-parameter tuned to find the best C values  $C(1.0, 0.01, 0.001, 0.000)$ , and if the weights should be balanced such that positive instances are weighted in proportion to how rare they are *balanced*(0, 1) (See Section

Unfortunately, doc2vec representations could not be obtained for the movies or place-types domains as the original full text was not available, only the bag-of-words.

<sup>8</sup><https://nlp.stanford.edu/projects/glove/>

# Re-organizing Vector Spaces into Interpretable Representations

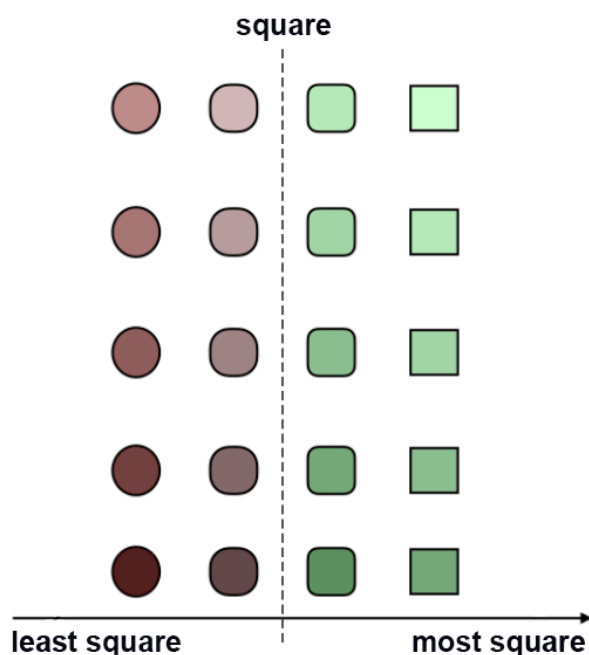
## 4.1 Introduction

Vector space models encode meaning spatially, but their features are typically entangled. This entanglement means that their features are not meaningful, limiting potential interpretability limits that could lead to off-the-shelf application in real-world domains like Medicine, the Criminal Justice System and Financial Markets (discussed in Section 2.7). However, they achieve strong results in a variety of domains and see widespread use as they are flexible in how they can be learned, e.g. by integrating word-context to achieve strong results on sentiment tasks [?], learning visual data alongside word-data to explain the content of images [?], and enforcing grammatical structure to perform better at question answering tasks [?].

This chapter is about re-structuring document embeddings (vector space models of documents) such that their spatial relationships are disentangled into an interpretable feature representation. To give insight into what kind-of features this method can obtain, the following example is given: where documents are concatenated movie reviews for a particular movie (See Section 3.2). In this domain, documents would be represented by features like "Scary", which would be how scary a movie is, or "Romantic" which would be how romantic a movie is.

This chapter follows work by Derrac[?], who first introduced the method to achieve this. The method begins with the following assumption: if documents in a document embedding can be linearly separated based on binary word occurrence (where a document is 1 if the word occurs and 0 otherwise), that word is semantically important in the domain. This can be achieved in an





**Figure 4.1:** An example of a hyper-plane in a toy domain of shapes. The hyper-plane for the word square is the dotted line. Green shapes are positive examples and red shapes are negative examples. Those closest to the hyper-plane are less square than those further away.

unsupervised way by training a linear model, e.g. a linear Support Vector Machine (SVM) (See Section ??), where the words that are most semantically important in the domain can be determined by evaluating how well the documents are separated, using standard model evaluation metrics like F1-score (see Section ??).

Linear SVMs obtain a hyper-plane for each model that separates documents that contain the word and documents that do not contain it. We show an example of this in a toy domain of shapes in 4.1, where the dotted line is a hyper-plane. As shown in this example, it can be assumed that documents furthest from the hyperplane on the negative side are the least representative of the thing being separated by the hyper-plane, in this case the 'squareness' of a shape, and the documents that are furthest from the hyper-plane on the positive side are the most representative, while those closest to the hyper-plane are more ambiguous. By simply taking the coefficients (orthogonal vector) of this hyper-plane, a direction can be obtained that goes from documents that are the most distant from the hyper-plane on the negative side, to those that are

most distant from the hyper-plane on the positive side (see the direction shown at the bottom of the example in 4.1).

By measuring how far up a document is on the orthogonal direction vector for a word, a ranking of that document on that word can be obtained, e.g. how 'Funny' it is relative to the other documents. After repeating this process for all documents on a word direction, features that rank documents on a variety of words can be obtained. To get a representation that contains less noise, only those words that are well spatially separated in the representation can be included. This is the first step of the approach described by ?? towards re-organizing a document embedding such that the features encoded spatially are disentangled into features of the representation. Specifically, first hyper-planes are obtained for all words based on binary frequency. Then, words are identified that are semantically important (by e.g. F1 Score or accuracy of the SVM classifier). Finally, orthogonal directions for these word hyperplanes are obtained and documents are ranked on how far up they are on these directions. These ranking of high-scoring words are then used as the new disentangled representation.

However, these word labels of directions, although direct, can be ambiguous. For example, in a domain of IMDB movie reviews "numbers" could be referring to musical "numbers" in a Broadway musical movie, or the amount of mathematics done by the actors. To resolve this, similar words can be clustered together e.g. we can give context to the word "numbers" by clustering together similar word directions "singing songs musical song numbers dance dancing sings sing Broadway". This can be done with an off-the-shelf clustering algorithm like K-means (see 4.4) that uses the word direction vectors as input. We show examples of these word cluster labels for the features of our interpretable representation in 4.1.

This Chapter builds on the original method to find directions in a document embedding model and rank documents on them introduced by Derrac [?]. Specifically, the contribution of this chapter is an extensive quantitative examination in section ?? and qualitative investigation in Section ?? across five domains (as described in Chapter 3) and four document embedding models. Variants to the method introduced in this work are specified in Section 4.5, with the overall goal of the chapter to examine how the new variants perform relative to each other, understand their degree of disentanglement by using simple decision tree classifiers, and verify that the method works in a variety of domains and document embedding models. Finally, conclusions are made on the contribution of the chapter in section ??.

IMDB Movie Reviews	Flickr-Placetypes	20-Newsgroups
courtroom legal trial court	broadway news money hollywood	switzerland austria sweden swiss
disturbing disgusting gross	fir bark activism avian	ham amp reactor watts
tear cried tissues tears	palace statues ornate decoration	karabag armenian karabakh azerbaijan
war soldiers vietnam combat	drummer produce musicians performers	4800 parity 9600 bps
message social society issues	ubahn railways electrical bahn	xfree86 linux
events accuracy accurate facts	winery pots manor winecountry	umpires umpire 3b viola
santa christmas season holiday	steeple religion monastery cathedral	atm hq ink paradox
martial arts kung	blanket whiskers fur adorable	lpt1 irq chipset mfm
bizarre weird awkward	desolate eerie mental loneliness	manhattan beauchaine bronx queens
drug drugs dealers dealer	carro shelby 1965 automobiles	photoshop adobe
inspirational inspiring fiction narrative	relax dunes tranquil relaxing	reboost fusion astronomers galactic

**Table 4.1: Example features from three different domains, where each cluster of words corresponds to a direction which movies are ranked on.**

## 4.2 The Use of Directions in Vector Spaces

Directions in document embedding models that go from documents that least represent a word, to those that most represent it, can be useful in a wide variety of applications. The most immediate example is perhaps that they allow for a natural way to implement critique-based recommendation systems, where users can specify how their desired result should relate to a given set of suggestions [?]. For instance, [?] propose a movie recommendation system in which the user can specify that they want to see suggestions for movies that are “similar to this one, but scarier”. If the direction of being scary is adequately modelled in a document embedding model of movies, such critiques can be addressed in a straightforward way. Similarly, in [?] a system was developed that can find “shoes like these but shinier”, based on a document embedding model that was derived from visual features. Semantic search systems can use such directions to interpret queries involving gradual and possibly ill-defined features, such as “popular holiday destinations in Europe” [?]. While features such as popularity are typically not encoded in traditional knowledge bases, they can often be represented as document embedding model directions. As another application, directions can also be used in interpretable classifiers. For example, [?] learned rule based classifiers from ranks induced by the feature directions.

Other work which has taken advantage of directions in vector spaces has relied on word-embeddings 2.4.10. For instance, [?] found that features of countries, such as their GDP, fertility rate or even level of CO<sub>2</sub> emissions, can be predicted from word embeddings using a linear regression model. In [?] directional vectors in word embeddings were found that correspond to

adjectival scales (e.g. bad < okay < good < excellent) while [?] found directions indicating lexical features such as the frequency of occurrence and polarity of words.

## 4.3 Background

## 4.4 Clustering

### 4.4.1 K-means

### 4.4.2 Derrac's K-means Variation

## 4.5 Method

This section details the methodology to disentangle a document embedding model starting with only itself and its associated Bag-Of-Words [?]. The work in this chapter differs from the method introduced by Derrac [?] as it focuses on investigating the quality of these disentangled representations using simple classifiers. Multiple variations are introduced and included as part of the experimental method.

### 4.5.1 Obtaining Directions and Rankings From Words

The method starts with a given document embedding model  $V_D$  induced from text documents  $d \in D$  and their associated bag-of-words  $B_D$ . For the bag-of-words  $B_D$  each document is composed of word frequencies  $d = (wf_1, wf_2, \dots, wf_n)$  where  $wf(d)$  is equal to the frequency of a word in a document and  $n$  is equal to the number of unique words in the vocabulary  $w \in W$ . Following the general explanation in the introduction, this section more precisely explains how to obtain a word-direction vector  $\mathcal{D}_w$  for all words in the vocabulary  $w \in W$ , by using a vector found by a linear model  $M_w$  that separates documents that have a word and do not have a word. Then, from that direction it explains how to obtain a ranking of all documents  $R_w = (rw_{d1}, rw_{d2}, \dots, rw_{dn})$  where  $rw_d$  is equal to the ranking of a document on a word direction

and  $n$  is the number of documents. The section following this one shows how to remove word directions that are not semantically important by evaluating the quality of the classifier that obtained the direction  $M_w$ , or the quality of the direction  $\mathcal{D}_w$ .

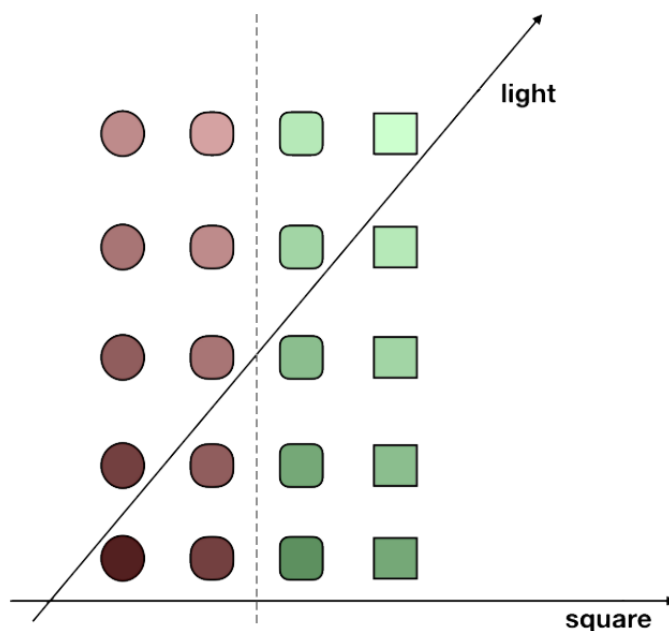
**Obtaining directions for each word** Each document is represented by a vector  $v_d$  in the document embedding model  $V_D$ . For this section, document vectors  $v_d$  are treated as points  $p_d$  in the document embedding model. For each word  $w$ , a hyper-plane  $h_w$  is obtained by training a linear model<sup>1</sup>  $M_w$  on the document embedding model  $V_D$  so that each document  $p_d$  in the document embedding model where the word  $w$  occurs more than once  $wf(d) \geq 1$  are separated from those where the word did not occur  $wf(d) = 0$ . We obtain such hyperplanes for all words in the vocabulary above a frequency threshold  $wf(D) > T$  where  $wf(D)$  is the frequency of the word in all documents. In practice, the parameter  $T$  is determined with hyper-parameter optimization. This task is unbalanced, i.e. there are typically fewer documents that contain the word compared to those that do not contain it, so the weights of the classifier are balanced such that positive instances are weighted in proportion to how rare they are.<sup>2</sup>

Although the hyperplane  $h_w$  is classifying a binary class (either classifying documents  $d_p$  as negative or positive), the distance between the document vectors  $d_p$  and the hyperplane  $h_w$  will vary. For example, when separating documents based on the occurrence of a word, it can be expected that the documents which contain the word more frequently would be further away from the hyper-plane on the positive side. We give an example of two directions in Figure 4.2. To apply this idea to a real domain, we can give an example from movie reviews, where the word is 'Scary' and the most 'Scary' movies are at the tip of the direction and those that are least 'Scary' are at the base of the direction. With this understanding, the direction  $\mathcal{D}_w$  can be obtained by simply taking the vector perpendicular to the hyperplane  $h_w$ . This direction goes from documents  $d_p$  from those lowest on the direction (at the distance furthest from the hyperplane on the side where documents  $d_p$  are classified) to those highest on the direction at the distance furthest from the hyperplane at the positive side.

**Ranking documents on directions** In this section we specify how to obtain a ranking  $R_w$  of all documents on a word direction vector  $\mathcal{D}_w$ . The rank of a document  $d$  can be defined by the dot product  $\mathcal{D}_w \cdot p_d$  as the ranking  $rw_d$  of the document  $d$  for the word  $w$ . Specifically,  $rw_{d_1}$  is

<sup>1</sup>Tested using a logistic regression classifier and a linear SVM, both achieved similar results

<sup>2</sup>Using scikit-learn, class\_weight:'balanced'



**Figure 4.2: Another example of a hyper-plane in a toy domain of shapes. Here we show multiple directions, one for light and one for square. The hyper-plane for the word square is the dotted line. Green shapes are positive examples and red shapes are negative examples for the word square. .**

ranked higher than  $rw_{d_2}$  if  $rw_{d_1} < rw_{d_2}$ . These rankings measure how relevant the document is in the spatial representation for the word, rather than just frequency e.g. a document that contains the word "scary" but isn't a scary movie (e.g. if it contained sentences like "it's scary how much money is spent on advertising movies like this") would not be ranked highly on the direction for 'scary', as the word 'scary' is not semantically important for the document. To put it another way, intuitively it can be understood to mean that the document  $d_2$  'has' the feature to a greater extent than  $d_1$ , e.g. in a domain of movie reviews if a movie ranked highly on the word 'dull', the movie has more dullness than lesser ranked movies.

In this section, the methodology to obtain word-directions and their associated rankings was described. These word-rankings are useful as features, and hypothetically we could obtain a representation that has as many features as there are words. However, some words are more semantically important in the domain than others. The next step describes how to remove word directions that are not well predicted by the linear model  $M_w$ , under the assumption that if they are not spatially important (i.e. easily separable), they are not semantically important. Another problem with word-rankings is that their meaning can be unclear, e.g. the word "serial" could

be referring to a series of movies, or a "serial" killer<sup>3</sup>. To solve this problem, the final section explains methods to cluster words together. This finally results in a representation where each feature is semantically important and has an associated cluster of words to give context. We gave examples of these clusters of words in the introductory table 4.1.

### 4.5.2 Filtering Word-Directions

Although we are able to obtain word directions for every word, not every word is semantically important in the document embedding model. Here, we distinguish between word-directions  $\mathcal{D}_w$  as directions that are not semantically important in the document embedding model, and feature-directions  $\mathcal{D}_f$ , which are. We define the set of feature directions as  $f \in F$ . Additionally, we make the distinction between a word-ranking  $R_w$  which is a ranking on a word-direction and a feature-ranking which is a ranking on a feature-direction. This section how to filter out word-directions so that only feature-directions remain, in-order for the final representation to be composed of only feature-rankings. Additionally, we refer to the word or cluster of words associated with a feature-ranking as a feature-label for that ranking.

The assumption made by Derrac in the original work [?] was that if a classifier  $M_w$  does not predict the occurrence of a word in a document in its embedding well, it is not semantically important. Put another way, if the documents are separated well, it must mean that the word  $w$  being used in the description of  $d$  is important enough to affect the document embedding model representation of  $d$ . The occurrence of a word in a document can be evaluated by using a variety of scoring metrics to determine the performance of the classifier  $M_w$ . This work also introduces a the use of a scoring metric that evaluates the quality of the direction  $\mathcal{D}_w$ , as even if the documents are well separable, then the ranking induced from the direction may not be correct. This metric compares how well the ranking induced by the hyperplane correlates with a BoW representation. If the ranking does correlate, it can be assumed that this means the word was strongly influential in the document embedding model, as the detail of the Bag-Of-Words information is embedded in the document embedding model's structure.

After scoring the words using one of the aforementioned metrics, a simple cut-off is applied

---

<sup>3</sup>The real cluster of words that this example comes from is "gore gory bloody blood gruesome serial investigate deaths"

where the top scoring words are taken as feature-directions (e.g. the top 2000 scored words). By obtaining these feature-directions, we can rank documents on each of them and use them as features of a representation  $I_{Fr}$  where  $Fr = (fr_{d1}, fr_{d2}, \dots, fr_{dn})$  and  $fr_d$  is the ranking of a document on a feature-direction. In this representation, each feature is semantically important, however there may be overlap, e.g. the word "Gore" and "Gory" likely have similar rankings. Ideally, the score cut-off would be at the point where the words stop corresponding to semantically important features. However, it is difficult to determine this, so in practice this value is taken as a hyper-parameter determined by a classifier on some domain task.

**Cohen's Kappa.** This is the only metric used in the work by Derrac [?]. This metric evaluates the performance of the classifier, and also deals with the problem that these words are often very imbalanced. For very rare words, a high accuracy might not necessarily imply that the corresponding direction is accurate, as if there are a large number of negative examples (as is the case with infrequent words) the classifier could simply predict that all documents do not contain the word to achieve a high score. For this reason, they proposed to use Cohen's Kappa score instead. In our experiments, however, it was found that this can be too restrictive, allowing us to sometimes obtain better results with the more simple accuracy metric.

**Classification accuracy.** If a model has high accuracy for a word  $w$ , it seems reasonable to assume that  $w$  describes a salient property for the given domain. However, despite balancing the weights of the original SVM used to obtain the hyper-plane, the value this metric places on correctly predicting negative classification compared to Kappa, it might favour rare words as it tends to be easier to obtain a high accuracy for these words.

**Normalized Discounted Cumulative Gain** This is the metric chosen to evaluate the quality of the rankings induced by the direction  $R_w$ . In-order to do so, two rankings are compared: one defined by the rankings and one defined by PPMI scores. The metric found to work best was Normalized Discounted Cumulative Gain (NDCG) which is a standard metric in information retrieval that evaluates the quality of a ranking w.r.t. some given relevance scores [?]. NDCG is mostly affected by the ranking position of the documents for which PPMI is highest. Spearman Rho, Gini, and Kendall Tau as alternative metrics do not favour higher ranked documents as much, but this comes with two problems. First, PPMI (See section ??) leads to a large number of zero scores. If we assume that all documents that have a zero frequency are ranked the same,



then the dot products rankings will be greatly different for lower-ranked documents as they instead are ranked according to their spatial representation. This disrupts the score too much to be useful when lower ranked documents are given equal importance to higher ranked ones. In this case, the rankings  $R_w$  of the document  $d$  are those induced by the dot products  $\mathcal{D}_w \cdot p_d$ . The relevance scores are determined by the Pointwise Positive Mutual Information (PPMI) score  $PPMI(w, d)$ , of the word  $w$  in the BoW representation of document  $d$  (See section ??), under the assumption that they correspond to a good baseline for what we consider to be important for an entity.

$$\begin{aligned} DCG_R^w &= \sum_{i=1}^{pr_d} \frac{ppmi_i^w}{\log_2(i+1)} \\ IDCGR^w &= \sum_{i=1}^{|documents|} \frac{2^{ppmi_i^w} - 1}{\log_2(i+1)} \\ nDCG_R^w &= \frac{DCG_R^w}{IDCGR^w} \end{aligned}$$

To define NDCG, we can first define Discounted Cumulative Gain (DCG), where  $prw_d$  is equal to a position in the ranking of documents on a direction  $\mathcal{D}_w$ , and  $ppmi_{p_i}^w$  is equal to the PPMI score for a word at position  $i$  in the ranking. Then, we can define the Ideal Discounted Cumulative Gain (IDCG), which is the best possible DCG for a position  $prw_d$ , where  $|documents|$  are the documents for the term ordered by their relevance up to position  $prw_d$ . nDCG is then simply the DCG normalized by the iDCG.

### 4.5.3 Clustering Features

A representation composed only of rankings of single words could be used, however that comes with two issues when classifying with a simple classifier. The first is that there may be too many dimensions, so a classifier like a Decision Tree (See Section 2.4.2) needs to be deep to classify well. The second is that it can sometimes be ambiguous what a feature-ranking means when it is labelled with only a single word, e.g. the word "courage" has an associated feature-direction, but what that feature direction represents can only be understood in the context of a cluster of similar words "courage students teaches student schools teacher teach classes practice training learning overcome conflict teaching" showing that it is about courageous teachers and students

overcoming challenges. The most naive way to obtain a cluster like this is to find similar word-directions:

**Similarity Labelling** As a way to add context to single word directions, for each single word direction  $w \in D_w$ , the cosine similarity is calculated between it and every other word direction. Then, the top  $n$  most-similar words are concatenated with the label. However, this does not reduce the amount of features or modify the direction. By labelling feature-directions like this, documents can be ranked on the single word directions and also have associated feature labels to give context.

Another approach is to cluster similar directions together, obtaining new cluster directions  $d_c = C(w_1, w_2, \dots, w_n)$  where  $d_c$  is the vector output of the clustering algorithm  $C$  for some words (used as a label)  $L(w_1, w_2, \dots, w_n)$  that have been chosen to be clustered together. From these new cluster directions, a new representation  $I_{Cr}$  is taken where each feature is a ranking of entities on a cluster direction  $Cr_i = (cr_{d1}, cr_{d2}, \dots, cr_{dn})$  where  $cr_d$  is the ranking of a document on a cluster feature-direction. This has the same benefits of the previous method in that the number of features are reduced and words can be given context when clustered together as a label for the feature. However, they also change the representation, e.g. two directions that describe a similar feature of movies when clustered together e.g. the feature-directions for the words "Bloody" and "Gorey" will result in a better performing feature overall. Both are words in movie reviews to describe how much blood a movie contains, so if these feature directions are averaged then the cluster direction can be used to produce a more balanced ranking for how much blood there is in films. Essentially, the cluster feature-direction could more accurately represent the semantics of a bloody film, compared to what is possible when considering either feature-direction individually. Finally, clustering also can obtain new properties by clustering directions, e.g. "Bloody" refers to how bloody a film is, but when clustered with "Bloody", "Scary" and "Horror", the new clustered direction now models the property of a horror movie more accurately.

On the other hand, its possible that when clustering many words together the cluster direction no longer represents a semantically important feature. For example given the associated label for a cluster direction  $\{Romance, Love\}$  and a cluster direction  $\{Bloody, Gorey\}$  the feature-direction for  $\{Cute\}$  is more relevant to the former rather than the latter, and has been used in reviews for romance movies. But it has also been used in reviews for movies containing cute

animals. This would make the new clustered direction  $\{Romance, Love, Cute\}$  perform worse at classifying the movie genre "Romance", but a bit better at classifying if a movie contains animals. It might thus be preferable to keep *Cute* in a separate cluster that represents animal movies rather than a cluster that represents romantic movies with cute animals. In the quantitative results, sometimes clustering performed worse than single directions, and not being able to find the clusters that correspond to the specific classes in question can be attributed as to why, as they may have been disrupted by clustering together words that disrupt a more important meaning. This is the reason for the hyper-parameter optimization of the clustering methods, and introducing standard K-means as a variant.

We will experiment with two different clustering methods: k-means and a variant of k-means that was proposed in the work by Derrac [?]:

**K-Means** K-means is a standard baseline clustering algorithm. In the experimental results, it was found that Derrac's variation relies too much on the initial choice of cluster centers, meaning if that key directions may be missed. Avoiding this is difficult without extensive and sometimes arbitrary hyper-parameter optimization. As K-Means has easy to tune hyper-parameters, it can find clusters that correspond to the task.

Given the input  $X$ , traditional k-means begins with  $K$  centers chosen uniformly at randomly from  $X$ . In this work we use the K-means++ variation [?], where the first center is chosen uniformly at random from  $X$  and the remaining centers are chosen with the following probability, where  $D(x)$  is the shortest distance from a data point  $x \in X$  to the closest center that has already been chosen:

$$\frac{D(x)^2}{\sum_{x \in X} D(x)^2} \quad (4.1)$$

After this initialization, the standard k-means process is followed. The distance between each input  $x$  and center  $c$  is calculated. In-order for the Euclidean distance used in k-means to be meaningful, the vector directions are normalized. In particular the relationship between them is:

$$\cos(\text{angle}) = 1 - \text{dist}/2 \quad (4.2)$$

Each input  $x$  is then assigned to its closest center  $c$ . Then, the centers are recomputed to be the mean of their assigned inputs. This process starting with the distance calculation is repeated until the centers do not change or a maximum number of iterations is reached (in this case, the default parameter of 300 iterations is used).

**Derrac’s K-Means Variation** This is the clustering method used in the work this method was introduced in [?]. As input to the clustering algorithm, it considers the  $N$  best-scoring candidate feature directions  $v_w$ . Hyper-parameter selection is performed by varying the scoring method or amount of candidate feature directions  $N$ . The main idea underlying their approach is to select the cluster centers such that (i) they are among the top-scoring candidate feature directions, and (ii) are as close to being orthogonal to each other as possible.

The output of this clustering algorithm is a set of cluster directions  $CD = C_1, \dots, C_K$ , where each cluster direction  $C$  has an associated label of words it is composed of  $L(w_1, w_2, \dots, w_n)$ . Note that these a feature representation is obtained for these cluster directions as described in the previous section. In the following, we will write  $C_j$  to denote the centers of the directions corresponding to the words in the cluster  $L_j$ , and  $w_d$  for the direction corresponding to the word in that cluster.

The first cluster is chosen by taking the top-scoring direction for the chosen scoring metric. Then, the new center  $C$  is selected by choosing the word-direction that is least similar to all of the current centers:

$$C = \min_{C_j \in CD} (\max_{w_i \in W_n} (\cos(w_i, v_{C_j}))) \quad (4.3)$$

Where  $W_n$  are the candidate word-directions, and  $CD$  is the current set of cluster centers. Once  $k$  clusters have been selected, each candidate direction  $w_i$  is added to its most similar cluster  $\max_{C_j \in CD} (\cos(w_i, C_j))$  and after adding that direction the cluster direction is updated to be the average of the current center and the newly added word  $AVG(C_j, w_i)$ . This continues until all candidate directions  $W_n$  are added to clusters. The cluster centers  $CD$  are taken as the final cluster centers.

## 4.6 Experimental Method

For each domain, we filter out terms that do not occur in at least two documents, and additionally limit the maximum number of words in a vocabulary to 100,000. For each task in each domain, each of the datasets are split into a 2/3 training data, 1/3 test data split. We additionally remove 20% of the training data and use that as development data for our hyper-parameters.

### 4.6.1 Document Embedding Models

Below the choices for the document embedding models that are formally described in Section ?? are explained:

## 4.7 Qualitative Results

### 4.7.1 The top-scoring directions for each domain

To give an understanding of the kind of directions found for each domain, the top-scoring ones are presented in Table 4.2. The document embedding model and scoring metric are determined by what performed best on an associated domain task when using the directions as input to a depth-3 decision-tree. For the movies, the task is classifying the Genre of the movie, for the Flickr tag place-types the task is classifying place-types according to the Foursquare taxonomy. Note that the table reports the best scoring directions, rather than the resulting clusters. However, for clarity, for each word, we also show the two words with the most similar direction. There is an interesting difference between the sentiment directions and the movies directions in table 4.2. Both of these domains are composed of movie reviews, but the documents in the former are a concatenation of a number of reviews across different sources, while the latter are individual reviews. This has resulted in the more general properties that apply to many movies being salient in the movies domain, but are less important than the names of actors and actresses in the sentiment domain.

In the movies domain, properties of movies e.g. genre related directions like "Horror" or "Romantic" are the highest scoring, meaning they are the most important spatially, and in-turn

semantically in the representation. However, in the sentiment domain the properties that are highest scored are names of actors and actresses. These names are important to distinguish between *reviews of movies*, but not between *movies themselves*. In more concrete terms, actor names are both rare and definitive when distinguishing between movie reviews. Similarly For the newsgroups domain, directions that are good at categorizing certain newsgroups are found, as the data is split into newsgroups e.g. the word 'celestial' applying to religious newsgroups, or the "diesel" and "porsche" directions relating to the newsgroup that discusses cars ("auto"). In the case of the place types domain, we generally find objects that occur in the photo "stream", "wilderness", "cliff", adjectives describing them "peaceful", "tranquil", or awards that are given to sub-categories of photos like "mygearandme" (a group competition where users submit photographs). As this is the representation that scored well on the Foursquare task (See 3.2), it is unsurprising that the important directions are in these categories, as the classes are types of places. In this case, the nature-related directions are probably useful for classifying the "Parks and Outdoors" class. The reuters dataset is certainly a case where providing context is important, as it is mostly business jargon ("quarterly", "avg", "dlr", "1st (qtr)").

### 4.7.2 Comparing Document Embedding Models

In this section, the differences between the different document embedding models for the movies domain are illustrated. However, as the original text was not available for this domain, the doc2vec embeddings are not available for this domain. In these examples, document embedding models of size 200 and score-type NDCG are used as they generally perform well. Additionally, the top 20,000 most frequent words are used when obtaining results for an embedding type. In terms of quantitative performance, when MDS is used as input to a Linear SVM or Decision Tree, it performs better than the others in F1-score (See in the Quantitative Results section, Table 4.8). Theoretically, this means that it should contain unique natural directions that other document embedding models do not have.

In Table 4.3 common and unique terms between the document embeddings are shown. A term is common if the term occurs in the 2,000 top scoring terms of both the other two embeddings. A term is unique if neither the direction name, or the direction names provided as context in brackets occur in any of the other top 2,000 scoring terms for the other embeddings. This is to

<b>Movies</b> (50 MDS NDCG)	<b>Sentiment</b> (100 D2V NDCG)	<b>News</b> (50 D2V NDCG)	<b>Place-types</b> (50 AWV Kappa)	<b>Reuters</b> (200 MDS NDCG)
horror (scars, scary)	glenda (glen, matthau)	karabag (iranian, turkiye)	hike (peak, climbing)	franklin (fund, mithly)
hilarious (funniest, hilarity)	scarlett (gable, dalton)	leftover (flaming, vancouver)	stream (waterfall, fountains)	quarterly (shearson, basis)
bollywood (hindi, india)	giallo (argento, fulci)	wk (5173552178, 18084tmibmclmsuedu)	arquitectura (edificio, arquitetura)	feb (28, splits)
laughs (funnier, funniest)	bourne (damon, cusack)	1069 (mlud, wibbled)	peaceful (serene, tranquil)	22 (booked, hong)
jokes (gags, laughs)	piper (omen, knightley)	providence (norris, ahl)	cathedral (roman, religious)	april (monthly, average)
comedies (comedic, laughs)	casper (dolph, damme)	celestial (interplanetary, bible)	architektur (architectuur, fenster)	sets (principally, precious)
hindi (bollywood, india)	norris (chuck, rangers)	mlud (wibbled, 1069)	landscapes (soe, flickrdiamond)	16 (creditor, trillion)
war (military, army)	holmes (sherlock, rathbone)	endif (olwm, ciphertxt)	stones (mygearandme, flickrbest)	1st (qtr, pennsylvania)
western (outlaw, unforgiven)	rouke (mickey, walken)	gd3004 (35894, intergraph)	wilderness (glacier, peak)	26 (approve, inadequate)
romantic (romance, chemistry)	ustinov (warden, cassavetes)	rtfnnmiedu (newsanswers, iee)	nationalpark (hike, peak)	23 (offsetting, weekly)
songs (song, tunes)	scooby (doo, garfield)	eng (padres, makefile)	tranquil (serene, peaceful)	prior (recapitalization, payment)
sci (science, outer)	doo (scooby, garfield)	pizza (bait, wiretap)	waterfall (butterfly, stream)	avg (shrs, shr)
funniest (hilarious, funnier)	heston (charlton, palance)	porsche (nanao, mercedes)	geology (mineral, formations)	june (july, venice)
noir (noirs, bogart)	homer (pacino, macy)	gebeadredslpittedu (n3jxp, skepticism)	serene (tranquil, peaceful)	march (31, day)
documentary (documentaries, footage)	welles (orson, kane)	scsi2 (scsi, cooling)	cliff (cliffs, rocky)	regular (diesel, petrol)
animation (animated, animators)	frost (snowman, damme)	playback (quicktime, xmotif)	naturesfinest (goose, ilovenature)	4th (qtr, fourth)
adults (adult, children)	streisand (bridget, salman)	35894 (gd3004, medin)	foliage (branch, blossom)	27 (chemlawn, theyre)
creepy (spooky, scary)	davies (rhys, marion)	diesel (volvo, shotguns)	dome (mosaic, column)	14 (borrowing, borrowings)
gay (gays, homosexuality)	cinderella (fairy, stepmother)	evolutionary (shifting, hulk)	baroque (neoclassical, renaissance)	11 (chapter, ranged)
workout (intermediate, instruction)	boll (uwe, belushi)	techniciandr (obp, 144k)	concert (guitar, performance)	may (probably, however)
thriller (thrillers, suspense)	rochester (eyre, dalton)	8177 (obp, 144k)	waves (500d, diamondclassphotographer)	38 (33, strong)
funnier (laughs, funniest)	edie (soprano, vertigo)	shaw (medicine, ottoman)	cliffs (cliff, rocky)	m1 (m2, m3)
suspense (suspenseful, thrillers)	scarerow (zombies, reese)	scorer (gilmour, lindros)	plaza (centro, streets)	dlr (writedown, debt)
arts (hong, chan)	kramer (strep, meryl)	xwd (xloadimage, openwindows)	flora (theunforgettablepictures, blueribbonwinner)	five (years, jones)
christianity (religious, religion)	marty (amitabh, goldie)	ee (275, xloadimage)	rocky (rugged, overlook)	bushels (soybeans, ccc)
musical (singing, sing)	columbo (falk, garfield)	com2 (com1, v32bis)	calm (serene, peaceful)	revs (net, 3for2)
gore (gory, blood)	kidman (nicole, jude)	examiner (corpses, brass)	branches (vivid, bushes)	29 (175, include)
animated (animation, cartoon)	juliet (romeo, troma)	migraine (ama, placebo)	mygearandme (platinumphoto, blueribbonwinner)	acquisition (make, usairs)
gags (jokes, slapstick)	garland (judy, lily)	parliament (parliamentary, armored)	stadt (ciudad, capitale)	payable (div, close)
sexual (sexually, sexuality)	hawn (goldie, matthau)	manhattan (bobbeviceicotekcom, beauchaine)	fauna (mammal, insect)	13 (dlrsbbl, groups)

Table 4.2: The top-scoring words for each domain.

ensure that unique directions are shown, rather than unique terms that refer to the same direction. The commonalities between document embedding model are much more prevalent than the differences, with natural properties of the domain being represented in all of the different document embedding models.

When examining the table of results, the common terms are mostly salient properties relevant to the domain, e.g. "noir", "comedies", "western", "documentary". The space learned using MDS captures the most unique properties (39 versus 31 from PCA), and also features some relevant to the domain that others do not have, e.g. "kung (martial, jackie)", "comics (comedian, comedians)", "kidnapping (kidnapped, torture)", "gambling (vegas, las)", despite also capturing some unique noise "berardinelli (employers, distributor)", "crawford (joan, davis)". The AUV space capture names, and properties which are interesting but minimal (train, slaves). Meanwhile PCA seems to contain many unique but noisy properties that are broadly in three categories, metadata from the review sites e.g. "copyright (*email, compuserve*)", "compuserve (*copyright, internetreviews*)", opinions related to the sentiment of the movie "negative (*positive, bother*)", "expressed (*reflect, opinions*)", "talents (*admit, agree*)" and words that are difficult to understand as they do not describe anything, but may be movie review jargon "stands (*fails, cover*)", intended (*bother, weren't*), "developed (*introduced, sounds*)".

### 4.7.3 Comparing Scoring Metrics

In Table 4.4 the same MDS document embedding as the previous section is used but the score-type is varied, in-order to examine the differences and commonalities between score-types. Similarly to the previous section, the most consistently meaningful properties are those that are common to all score-types, e.g. "horror (*scares, scares*)", "laughs (*funnier, funnier*)", "thriller (*thrillers, thrillers*)". The top 2,000 NDCG terms performed best when used as input to a classifier on the genres task. Following this, it can be seen that a lot of new properties are introduced in NDCG compared to the other scoring types e.g. "gay (*homosexuality, sexuality*)", "satire (*parody, parodies*)", "marry (*married, marriage*)". This is quite different to the previous section, where despite MDS being the better performing document embedding type, it only contained some unique but meaningful terms. The top directions scored on the F1 metric are by and large seems is difficult to understand, referring to names or specific aspects of the scene,



MDS	AWV	PCA	Common
berardinelli ( <i>employers, distributor</i> )	billy ( <i>thrown, dirty</i> )	amount ( <i>leaving, pick</i> )	noir ( <i>fatal, femme</i> )
crawford ( <i>joan, davis</i> )	brother ( <i>brothers, boys</i> )	fails ( <i>fit, pick</i> )	gay ( <i>homosexual, homosexuality</i> )
hitchcocks ( <i>hitchcock, alfred</i> )	fonda ( <i>henry, jane</i> )	pick ( <i>fails, fit</i> )	prison ( <i>jail, prisoners</i> )
warners ( <i>warners, bros</i> )	building ( <i>built, climax</i> )	stands ( <i>fails, cover</i> )	arts ( <i>rec, robomod</i> )
nuclear ( <i>weapons, soviet</i> )	train ( <i>tracks, thrown</i> )	surprisingly ( <i>offer, fit</i> )	allens ( <i>woody, allen</i> )
joan ( <i>crawford, barbara</i> )	slaves ( <i>slavery, excuse</i> )	copyright ( <i>email, compuserve</i> )	jokes ( <i>laughs, joke</i> )
kidnapped ( <i>kidnapping, torture</i> )		length ( <i>reflect, expressed</i> )	animation ( <i>animated, cartoon</i> )
hop ( <i>hip, rap</i> )		profanity ( <i>reflect, producers</i> )	sherlock ( <i>holmes, detective</i> )
kung ( <i>martial, jackie</i> )		compuserve ( <i>copyright, internetreviews</i> )	western ( <i>westerns, wayne</i> )
ballet ( <i>dancers, dancer</i> )		talents ( <i>admit, agree</i> )	songs ( <i>song, lyrics</i> )
gambling ( <i>vegas, las</i> )		admit ( <i>agree, talents</i> )	comedies ( <i>comedy, laughs</i> )
alcoholic ( <i>drunk, alcoholism</i> )		developed ( <i>introduced, sounds</i> )	workout ( <i>exercise, challenging</i> )
waves ( <i>surfing, wave</i> )		intended ( <i>bother, weren't</i> )	laughs ( <i>funnier, hilarious</i> )
jaws ( <i>jurassic, godfather</i> )		constantly ( <i>putting, sounds</i> )	drug ( <i>drugs, addict</i> )
jungle ( <i>natives, island</i> )		tired ( <i>anyone, mediocre</i> )	sci ( <i>science, fiction</i> )
employers ( <i>berardinelli, distributor</i> )		produced ( <i>spoiler, surprising</i> )	documentary ( <i>documentaries, interviews</i> )
pot ( <i>weed, stoned</i> )		involving ( <i>believes, belief</i> )	students ( <i>student, schools</i> )
canadian ( <i>invasion, cheap</i> )		anymore ( <i>continue, tired</i> )	thriller ( <i>thrillers, suspense</i> )
murphy ( <i>eddie, comedian</i> )		leaving ( <i>fit, pick</i> )	allen ( <i>woody, allens</i> )
comics ( <i>comedian, comedians</i> )		makers ( <i>producers, aspects</i> )	funniest ( <i>hilarious, laughing</i> )
kidnapping ( <i>kidnapped, torture</i> )		introduced ( <i>developed, considered</i> )	gags ( <i>jokes, slapstick</i> )
subscribe ( <i>email, internetreviews</i> )		loses ( <i>climax, suffers</i> )	adults ( <i>children, adult</i> )
vegas ( <i>las, gambling</i> )		negative ( <i>positive, bother</i> )	animated ( <i>animation, cartoon</i> )
distributor ( <i>berardinelli, employers</i> )		expressed ( <i>reflect, opinions</i> )	dancing ( <i>dance, dances</i> )
wave ( <i>waves, surfing</i> )		mildly ( <i>mediocre, forgettable</i> )	teen ( <i>teenage, teens</i> )
rhodes ( <i>internetreviews, email</i> )		helped ( <i>putting, allowed</i> )	soldiers ( <i>soldier, army</i> )
hippie ( <i>pot, sixties</i> )		reflect ( <i>expressed, opinions</i> )	indie ( <i>independent, festival</i> )
weed ( <i>pot, stoned</i> )		opinions ( <i>reflect, expressed</i> )	suspense ( <i>suspenseful, thriller</i> )
caribbean ( <i>pirates, island</i> )		frequently ( <i>occasionally, consistently</i> )	creepy ( <i>scary, eerie</i> )
eddie ( <i>murphy, comedian</i> )		content ( <i>agree, proves</i> )	italian ( <i>italy, spaghetti</i> )
sixties ( <i>beats, hippie</i> )		allowed ( <i>helped, weren't</i> )	jews ( <i>jewish, nazis</i> )
... 8 More		suffers ( <i>lacks, loses</i> )	... 1480 more

Table 4.3: Unique terms between document embedding models

NDCG	F1	Accuracy	Kappa	Common
gay ( <i>homosexuality, sexuality</i> )	company ( <i>sell, pay</i> )	kennedy ( <i>republic, elected</i> )	definitely ( <i>alot, awesome</i> )	horror ( <i>scares, scares</i> )
arts ( <i>hong, chan</i> )	street ( <i>city, york</i> )	bags ( <i>listened, salvation</i> )	guns ( <i>gun, shoot</i> )	laughs ( <i>funnier, funnier</i> )
sports ( <i>win, players</i> )	red ( <i>numerous, fashion</i> )	summers ( <i>verge, medieval</i> )	flawless ( <i>perfection, brilliantly</i> )	jokes ( <i>gags, gags</i> )
apes ( <i>remembered, planet</i> )	project ( <i>creating, spent</i> )	revolve ( <i>sincerely, historian</i> )	mail ( <i>reviewed, rated</i> )	comedies ( <i>comedic, comedic</i> )
german ( <i>germans, europe</i> )	mark ( <i>favor, pull</i> )	locale ( <i>foster, sharply</i> )	garbage ( <i>crap, horrible</i> )	sci ( <i>scifi, alien</i> )
satire ( <i>parody, parodies</i> )	lady ( <i>actress, lovely</i> )	cooler ( <i>downward, reports</i> )	featurette ( <i>featurettes, extras</i> )	funniest ( <i>hilarious, hilarious</i> )
band ( <i>rock, vocals</i> )	fire ( <i>ground, force</i> )	spades ( <i>ralph, medieval</i> )	complaint ( <i>extra, added</i> )	creepy ( <i>spooky, spooky</i> )
crude ( <i>offensive, offended</i> )	post ( <i>essentially, purpose</i> )	filmography ( <i>ralph, experiments</i> )	mission ( <i>enemy, saving</i> )	thriller ( <i>thrillers, thrillers</i> )
dancing ( <i>dance, dances</i> )	heads ( <i>large, throw</i> )	quentin ( <i>downward, anime</i> )	ruin ( <i>wondering, heck</i> )	funnier ( <i>laughs, laughs</i> )
restored ( <i>print, remastered</i> )	water ( <i>land, large</i> )	employers ( <i>finishes, downward</i> )	wars ( <i>forces, enemy</i> )	suspense ( <i>suspenseful, suspenseful</i> )
drugs ( <i>drug, abuse</i> )	road ( <i>drive, trip</i> )	formal ( <i>victory, kennedy</i> )	prefer ( <i>compare, added</i> )	gore ( <i>gory, gory</i> )
church ( <i>religious, jesus</i> )	brother ( <i>son, dad</i> )	tube ( <i>esta, muscle</i> )	heroes ( <i>packed, hero</i> )	gags ( <i>jokes, jokes</i> )
sexuality ( <i>sexual, sexually</i> )	party ( <i>decide, hot</i> )	woefully ( <i>restless, knockout</i> )	necessarily ( <i>offer, draw</i> )	science ( <i>sci, sci</i> )
sexually ( <i>sexual, sexuality</i> )	badly ( <i>awful, poorly</i> )	scientists ( <i>hilarity, locale</i> )	portray ( <i>portrayed, portraying</i> )	gory ( <i>gore, gore</i> )
england ( <i>british, english</i> )	limited ( <i>aspect, unlike</i> )	overboard ( <i>civilized, chiderella</i> )	critic ( <i>reviewed, net</i> )	government ( <i>political, political</i> )
ocean ( <i>sea, boat</i> )	impression ( <i>instance, reasons</i> )	rumors ( <i>homosexuality, characteristics</i> )	reviewed ( <i>rated, mail</i> )	suspenseful ( <i>suspense, suspense</i> )
marry ( <i>married, marriage</i> )	trip ( <i>journey, road</i> )	salvation ( <i>bags, cooler</i> )	saving ( <i>carry, forced</i> )	frightening ( <i>terrifying, terrifying</i> )
campy ( <i>cult, cheesy</i> )	michael ( <i>producers, david</i> )	actively ( <i>assassination, overcoming</i> )	technical ( <i>digital, presentation</i> )	military ( <i>army, army</i> )
christian ( <i>religious, jesus</i> )	memory ( <i>forgotten, memories</i> )	stretching ( <i>victory, hideous</i> )	statement ( <i>exist, critical</i> )	slapstick ( <i>gags, gags</i> )
melodrama ( <i>dramatic, tragedy</i> )	james ( <i>robert, michael</i> )	downward ( <i>cooler, crawling</i> )	shocked ( <i>hate, warning</i> )	scary ( <i>scare, scare</i> )
sing ( <i>singing, sings</i> )	thin ( <i>barely, flat</i> )	rocked ( <i>staple, demented</i> )	flying ( <i>air, force</i> )	blu ( <i>unanswered, ray</i> )
sentimental ( <i>touching, sappy</i> )	pre ( <i>popular, include</i> )	affectionate ( <i>esta, muscle</i> )	danger ( <i>dangerous, edge</i> )	internetreviews ( <i>rhodes, rhodes</i> )
depressing ( <i>bleak, suffering</i> )	faces ( <i>constant, unlike</i> )	protest ( <i>protective, assassination</i> )		cgi ( <i>computer, computer</i> )
evidence ( <i>investigation, accused</i> )	values ( <i>exception, wise</i> )	confined ( <i>cooler, downward</i> )		email ( <i>web, web</i> )
adorable ( <i>cute, sweet</i> )	unusual ( <i>odd, seemingly</i> )	inhabit ( <i>quentin, drawback</i> )		thrilling ( <i>thrill, exciting</i> )
episodes ( <i>episode, television</i> )	lovers ( <i>lover, lovely</i> )	latin ( <i>communities, mount</i> )		web ( <i>email, email</i> )
teenager ( <i>teen, teenage</i> )	frame ( <i>image, effect</i> )	reception ( <i>como, finishes</i> )		horror ( <i>scares, scares</i> )
magical ( <i>fantasy, lovely</i> )	mans ( <i>ultimate, sees</i> )	uptight ( <i>suspensful, stalked</i> )		laughs ( <i>funnier, funnier</i> )
health ( <i>medical, suffering</i> )	efforts ( <i>generally, nonetheless</i> )	brink ( <i>inexplicable, freddy</i> )		suspense ( <i>suspenseful, suspenseful</i> )

Table 4.4: Different score types

e.g. "company (*sell, pay*)", "post (*essentially, purpose*)", "impression (*instance, reasons*)", and accuracy is similar, e.g. "bags (*listened, salvation*)", "summers (*verge, medieval*)", "woefully (*restless, knockout*)" is similar. Kappa has some unique sentiment related terms e.g. "flawless (*perfection, brilliantly*)", "shocked (*hate, warning*)" "garbage (*crap, horrible*)" but also seems to contain some metadata e.g. "featurette (*featurettes, extras*)", "critic (*reviewed, net*)", "reviewed (*rated, mail*)", and although it has a couple of meaningful terms e.g. "guns (*gun, shoot*)", "flying (*air, force*)" it does not contain unique meaningful and general properties the way NDCG does. These qualitative examples contribute to the idea that NDCG, by evaluating the ranking rather than the separability, ends up discovering unique properties that are applicable to general domain tasks.

### Investigating Doc2Vec In The Newsgroups Domain

The previous examples were from the movies domain, however this did not include the doc2vec representation. For that reason, the 20 newsgroups domain is used to investigate the doc2vec space. In Table a comparison is shown between an MDS space, used as a representative of document embedding representations that use Positive Pointwise Mutual Information as their initial data input, and doc2vec that is a distributional representation, using word-context in its training. Doc2vec has achieved strong performance on a variety of baselines [?]. The aim of this analysis is to explore whether word-vectors and word-context can find interesting unique directions compared to MDS obtained from a PPMI BOW. In general, it is found that MDS contains a lot more noisy properties than D2V, specifically related to insignificant conversational words, e.g. "hi (*folks, everyone*)", "sorry (*guess, hear*)" "say (*nothing, anything*)" that do not seem related to capturing e.g. information that can separate entities into their original newsgroups. It seems that doc2vec was better at recognizing these words as noise and uninteresting compared to MDS, which must have prioritized these words. Doc2Vec represents interesting unique properties, e.g. "cryptology (*attendees, bait*)", which is very relevant to the newsgroup topic of cryptography. As expected, the common words are relevant to the task. It can be expected that by using word vectors, Doc2Vec is able to more easily identify interesting words and de-prioritize words which are common to the English language despite potentially being more rare in a smaller dataset.

D2V	MDS	Common
leftover (pizza, brake)	hi (folks, everyone)	chastity (shameful, soon)
wk (5173552178, 18084tmibmclmsuedu)	looking (spend, rather)	n3jxp (gordon, gebcadredslpittedu)
eng (padres, makefile)	need (needs, means)	skepticism (gebcadredslpittedu, n3jxp)
porsche (nanao, 1280x1024)	post (summary, net)	anyone (knows, else)
diesel (cylinders, steam)	find (couldnt, look)	gebcadredslpittedu (soon, gordon)
scorer (gilmour, lindros)	hello (kind, thank)	intellect (soon, gordon)
parliament (caucasus, semifinals)	david (yet, man)	please (respond, reply)
atm (padres, inflatable)	got (mine, youve)	thanks (responses, advance)
cryptology (attendees, bait)	go (take, lets)	email (via, address)
intake (calcium, mellon)	question (answer, answered)	know (let, far)
433 (366, 313)	interested (including, products)	get (wait, trying)
ghetto (warsaw, gaza)	list (mailing, send)	think (important, level)
lens (lenses, ankara)	sorry (guess, hear)	good (luck, bad)
rushdie (sinless, wiretaps)	heard (ever, anything)	shafer (dryden, nasa)
immaculate (porsche, alice)	cheers (kent, instead)	bobbeviceicetekcom (manhattan, beauchaine)
keenan (lindros, bosnian)	say (nothing, anything)	dryden (shafer, nasa)
boxer (jets, hawks)	number (call, numbers)	im (sure, working)
linden (mogilny, 176)	mailing (list, send)	sank (bronx, away)
candida (yeast, noring)	call (number, phone)	banks (soon, gordon)
octopus (web, 347)	thank (thanx, better)	like (sounds, looks)
czech (detectors, kuwait)	read (reading, group)	shameful (soon, gordon)
survivor (warsaw, croats)	phone (company, number)	could (away, bobbeviceicetekcom)
5173552178 (circumference, wk)	mail (send, list)	would (appreciate, wouldnt)
18084tmibmclmsuedu (circumference, wk)	doesnt (isnt, mean)	beauchaine (bobbeviceicetekcom, away)
3369591 (circumference, wk)	lot (big, little)	ive (seen, never)
mcwilliams (circumference, wk)	thats (unless, youre)	surrender (soon, gebcadredslpittedu)
coldblooded (dictatorship, czech)	believe (actually, truth)	problem (problems, fix)
militia (federalist, occupying)	youre (unless, theyre)	windows (31, dos)
cbc (ahl, somalia)	send (mail, mailing)	gordon (soon, gebcadredslpittedu)

**Table 4.5: Comparing an MDS document embedding model to a D2V document embedding model for Newsgroups, where a D2V document embedding model performed best..**

4.7.3

## 4.8 Quantitative Results

### 4.8.1 Evaluation Method

The aim of this quantitative evaluation is to verify the disentanglement of the property feature-representations. Additionally this section serves to determine the hyper-parameters that perform best on the associated domain tasks for the document embedding models, the feature representation composed of rankings of entities on single-term directions, and the feature representation composed of rankings on cluster directions.

First some baseline results are obtained using the unsupervised representations as input to a variety of classifiers. Then, the disentangled feature representations are used as input to low-depth CART Decision Trees are used (See Background Section 2.4.2) limited to a depth of one, two or three. The idea behind this is that if the properties that were spatially encoded in the document embeddings are disentangled, then the decision trees with a limited depth should be able to perform close to or equal performance to the original results for the unsupervised representation. If only the the performance of the disentangled feature representation was being evaluated, Linear SVM's could be used. However, the aim of this section is not to prove that these disentangled feature representations perform well, but rather that they are disentangled.

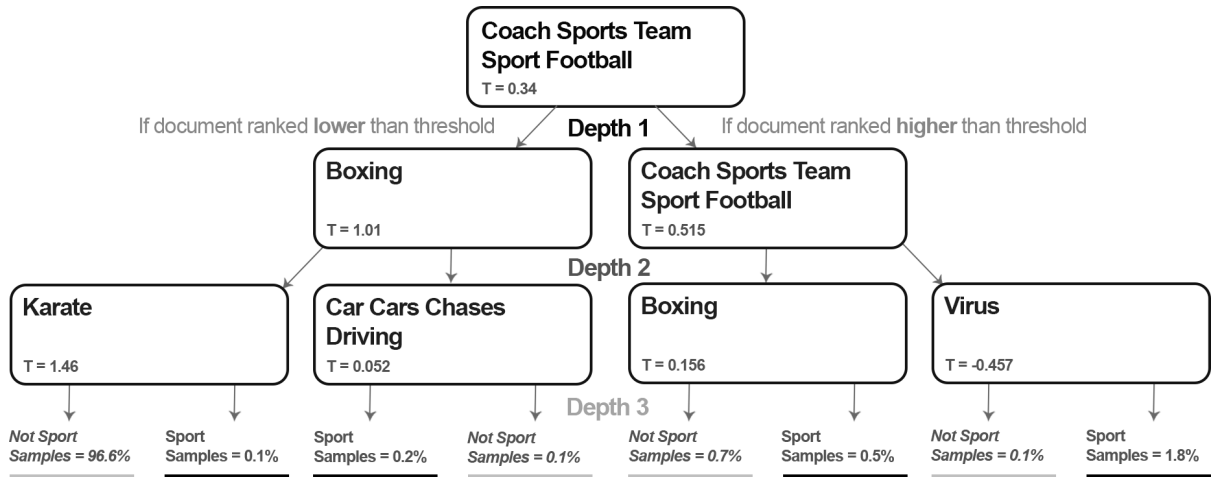
Essentially, if depth-one decision trees (that uses only a single feature to make a classification decision) can model a class well using the disentangled representation as input, then the information in the original document embedding that enabled it to perform well on the task has been disentangled into a single feature. Similarly, if the disentangled representation performs well on a depth-three limited decision tree, then it is safe to assume the method has disentangled a few different relevant properties from the original representation. Figure 4.3 shows one of the trees that was obtained from our feature based representation. If the document embedding is indeed disentangled into a feature representation composed of important properties for the domain, then these low-depth decision trees should generalize well to test data. To show the difference in entanglement between the original unsupervised document embeddings and the disentangled feature representations, low-depth decision tree results are also obtained for the unsupervised document embeddings.

### 4.8.2 Hyper-Parameters

As stated previously (See Section ??) in all experiments the document embedding models that results are obtained for are Doc2Vec, Principal Component Analysis (PCA), Multi-Dimensional Scaling (MDS) and Averaged Word Vectors (AWV). As possible choices for the number of dimensions, we used  $dim(50, 100, 200)$ . The Doc2Vec<sup>4</sup> implementation from the gensim library is used.

---

<sup>4</sup>Doc2Vec implemented in gensim



**Figure 4.3:** An example of a Decision Tree classifying if a movie is in the "Sports" genre. Each Decision Tree Node corresponds to a feature, and the threshold  $T$  is the required ranking of a document on that feature to traverse right down the tree instead of left. One interesting point to note is that the most important direction is used twice, the "coach, sports, team, sport, football" cluster and results in a majority of negative samples. Another point is that the nodes at depth three are more specific, sometimes overfitting (e.g. in the case of the "Virus" node, likely overfitting to a single movie about a virus) .

### Classifier Parameters

There are two classifiers that are used across all experiments. CART Decision Trees<sup>5</sup> and linear SVM's<sup>6</sup>.

Each SVM is also hyper-parameter tuned to find the best  $C$  values  $C(1.0, 0.01, 0.001, 0.000)$ , the best gamma value  $gamma(1.0, 0.01, 0.001, 0.000)$  and if the weights should be balanced such that positive instances are weighted in proportion to how rare they are  $balanced(0, 1)$ <sup>7</sup>. In particular, where  $n$  is the number of documents, and  $y$  is the number of documents that are classified positively by a binary class, the weight of a class is equal to:

$$\frac{n}{2 * y} \quad (4.4)$$

Additionally, each Decision Tree is hyper-parameter tuned for the following parameters:

<sup>5</sup>Decision Tree implemented in scikit-learn

<sup>6</sup>Implemented in scikit-learn

<sup>7</sup>Weights implemented in scikit-learn

- The number of features<sup>8</sup> to consider when looking for the best split. (*none*, *auto*, *log2*), where *none* means that all  $n$  features are considered, *auto* means that the maximum features considered is  $\sqrt{n}$ , and *log2* which means that the number of features considered is  $\log_2(n)$
- The scoring criterion for a node split *criterion* : *gini*, *entropy*. Where *gini* is the gini impurity and *entropy* is the information gain, see Section 2.4.2 for more detail.
- If the weights should be balanced *balanced*(0, 1)

A parameter in the decision tree that is not tuned but is used is the maximum depth.

### Experiment One: Obtaining baselines

These are three different steps of experimentation. The first experiment is to determine the best document embedding models for performance when used as input to depth-one, depth-two, depth-three and unbounded decision trees, as well as a linear SVM. This provides a frame of reference for the later experiments for disentanglement in the case of the depth-limited trees and baseline performance on the task in the case of the linear SVM. In addition to the unsupervised document embedding models, two additional document models are included as reference: a bag-of-words of PPMI scores (BOW-PPMI) and a Latent Dirichlet Allocation (LDA) topic model. The BOW-PPMI is used as a reference for a standard performance baseline representation on the task, and the LDA topic model is used as a reference interpretable representation (where the information in the unstructured text is separated into topics). After the original filtering done to the vocabulary where either only the top 100,000 most frequent terms are used, or those that did not occur more than once are removed, the BOW-PPMI additionally has all terms filtered out that do not occur in at least 0.1% of documents. This is to limit the memory requirements of the representation, but also to ensure that noisy terms that can be overfit on are not included as features. To determine the best parameters for Doc2Vec, the following parameters are tuned:

- The *window\_size*(5, 10, 15) referring to the context window of the words that used during training.

---

<sup>8</sup>"max\_features" in scikit-learn

- The  $\text{mincount}(1, 5, 10)$  referring to the minimum frequency of words included during training.
- The number of times the data is iterated on when training  $\text{epochs}(50, 100, 200)$ .

The best doc2vec embedding for each task is the one that performs best when used as input to a linear SVM. The idea is that if the representation can be separated for the key domain tasks with a linear SVM then it should contain informative directions. Additionally, the Topic Models need to be tuned. For the Latent Dirichlet Allocation topic models<sup>9</sup>, the following parameters are tuned:

- The prior of the document topic distribution  $(0.001, 0.01, 0.1)$ .
- The prior of the topic word distribution  $(0.001, 0.01, 0.1)$ .

### Experiment Two: The best single-term property features

The objective of this experiment is to verify that the single-term property features are disentangled. It may be the case that clustering directions does more harm to the feature representation than good, so these experiments are used to investigate the use of a feature representation composed only of rankings of entities on single terms. This is done-so by using feature representations composed of single term property rankings as input to low-depth Decision Trees. This experiment also finds the best-performing parameters on the task for the single-directions. The end-result is that for each task, there will be a best performing document embedding model and size of that model (e.g. Doc2Vec of size 50), and the following hyper-parameters are optimized for that model:

- The frequency cut-off of the terms  $f(20000, 10000, 5000)$ , where directions are only obtained for terms in the top  $f$ .
- The score cut-off of the directions  $s(2000, 1000)$ , where the feature-representation used as input to the classifiers is composed of rankings of entities on the top  $s$  directions.

---

<sup>9</sup>Topic models implemented in scikit-learn



- The score-type  $t$  ( $F1$ ,  $Kappa$ ,  $Accuracy$ ,  $NDCG$ ) which is the method used to score the directions.

The reason these hyper-parameters are optimized are because even the top-scoring directions are not always meaningful, as observed in the qualitative analysis in Table 4.4 for example.

### Experiment Three: The best cluster features

The objective of this experiment is to verify that the cluster features are disentangled. Clustering is an alternative method to using single-term directions that combines the directions into clusters. This results in a lower amount of linearly combined features labelled with clusters of words. As in the last experiment, this is done by using a feature-representation composed of cluster-rankings as input to low-depth Decision Trees. The two different clustering methods, K-means<sup>10</sup> and Derrac’s k-means variation (as described in Section 4.5.3), are used. The best single-term property parameters are used as the basis of this experiment, where for each classifier and task the clusters are obtained from the top  $s$  directions and score-type  $t$ , chosen by hyper-parameter optimization for that classifier and task.

The Derrac clustering method does not have hyper-parameters, but the following hyper-parameters are tuned for K-means:

- The number of times the algorithm will be run with different centroid seeds (5, 10, 50) where the final output is the run with the best inertia.
- The maximum number of iterations for a single run (100, 300, 1000)
- The tolerance (0.001, 0.0001, 0.00001, 0.0)

### 4.8.3 Summary of all Results

Table 4.8.4 is a summary of the three experiments, listing results of different representations on trees limited to depth-one, two and three. As expected, the original entangled unsupervised document embeddings did not perform well on this task. In contrast, single-directions and clusters

<sup>10</sup>K-means implemented in scikit-learn

of these single-directions obtained from these document embedding models out-perform the bag-of-words in most cases, with two exceptions being the depth-two tree for the place-types domain and the depth-one tree on the movies keywords task.

For the keywords task, in a depth-1 tree, finding words that directly correspond to particular keywords is simple with the BOW-PPMI representation. As the words that are able to classify these keywords will typically be infrequent (e.g. "new york city", "father-son relationship"), they are likely not as well-represented spatially in the document embedding model. In this case, the PPMI representation is suitable, as it can find 1-1 matches with the classes without modelling similarity information. When using depth-two and depth-three limited decision trees, the score is no-longer as good for PPMI, which is likely due to overfitting.

Sometimes Decision Trees of depth-two outperform those of depth-one, but generally depth-three trees perform best. In the case of the place-types, although topic models and PPMI representations are indeed the best, it is not by a wide-margin. Meanwhile when the single directions perform the best in these domains for other tree types they perform much better than the other approaches. Additionally, place-types has the least entities for its tasks and the tasks are very unbalanced, so it is possible that they overfit.

#### 4.8.4 Baseline Representations

In Table 4.7 Decision Trees and SVM results are shown when the document embeddings, the baseline representation BOW-PPMI, and the topic model features are used as input. In the case of the topic model and doc2vec, these representations are tuned. These representations serve as a reference point for what is possible using standard linear models, as well as a measure of the disentanglement of the original unsupervised document embeddings. The unsupervised document embeddings when used as input representations to depth three trees perform significantly worse in depth one trees. This is expected given the fact that these representations are entangled, so it is unlikely that a smaller tree can use the available dimensions to model a class. There are two tables, Table 4.7 that covers the results for the newsgroups in full detail (including more parameters, and the precision and recall scores), and Table 4.8 that shows results for all other domains but only shows the best document embeddings out of the different sizes. Note that in Table 4.7 some recall scores are very high. Recall scores are higher when the weights are

Movies	Genres			Keywords			Ratings		
	D1	D2	D3	D1	D2	D3	D1	D2	D3
Space	0.301	0.358	0.354	0.185	0.198	0.201	0.463	0.475	0.486
Single directions	<b>0.436</b>	0.463	0.492	0.23	<b>0.233</b>	<b>0.224</b>	0.466	0.499	0.498
Clusters	0.431	<b>0.513</b>	<b>0.506</b>	0.215	0.22	0.219	<b>0.504</b>	<b>0.507</b>	<b>0.513</b>
PPMI	0.429	0.443	0.483	<b>0.243</b>	0.224	0.224	0.47	0.453	0.453
Topic	0.415	0.472	0.455	0.189	0.05	0.075	0.473	0.243	0.38
	Newsgroups			Sentiment			Reuters		
Rep	0.251	0.366	0.356	0.705	0.77	0.773	0.328	0.413	0.501
Single dir	0.418	<b>0.49</b>	<b>0.537</b>	0.784	0.814	<b>0.821</b>	<b>0.678</b>	<b>0.706</b>	0.72
Cluster	0.394	0.433	0.513	0.735	<b>0.844</b>	0.813	0.456	0.569	0.583
PPMI	0.33	0.407	0.444	0.7	0.719	0.73	0.616	0.699	<b>0.723</b>
Topic	<b>0.431</b>	0.423	0.444	<b>0.79</b>	0.791	0.811	0.411	0.527	0.536
	Foursquare			OpenCYC			Geonames		
Placetypes	D1	D2	D3	D1	D2	D3	D1	D2	D3
Rep	0.438	0.478	0.454	0.383	0.397	0.396	0.349	0.34	0.367
Single dir	<b>0.541</b>	0.498	<b>0.531</b>	0.404	<b>0.428</b>	0.39	<b>0.444</b>	<b>0.533</b>	<b>0.473</b>
Cluster	0.462	0.507	0.496	<b>0.413</b>	0.42	<b>0.429</b>	0.444	0.458	0.47
PPMI	0.473	<b>0.512</b>	0.491	0.371	0.351	0.352	0.361	0.301	0.242
Topic	0.488	0.433	0.526	0.365	0.271	0.313	0.365	0.3	0.219

Table 4.6: The F1-scores of trees limited to depth-one, two and three, for the best hyper-parameter variation of each representation

balanced, as favouring positive instances results in more positives being identified correctly.

As seen in Table 4.7, the biggest differences in F1-score occur when the document embedding model changes, rather than when the size of document embedding model changes. For this reason, and to reserve space, only the best performing representation of each type are shown in Table 4.7.

In Table 4.7 for the newsgroups, the linear SVM only achieved strong performance when using doc2vec or PCA as input, and in the unbounded depth decision tree PCA performed far better than the other unsupervised document embeddings. Similarly for reuters and sentiment, the PCA representation performs better than the other unsupervised document embeddings. This suggests that the linear combinations used in the PCA representation result in a more linearly separable space. The performance gap only widens when looking at low-depth decision trees that use the PCA dimensions as input, which suggests that the individual dimensions of the other unsupervised embeddings are less informative than those of the PCA spaces. This is likely due to the fact that PCA tries to select the most informative dimensions, whereas the dimensions of the other representations are more or less arbitrary. However, this does not tell us anything about the quality of the directions that can be learned in both types of spaces.

In the single directions results, PCA is outperformed by MDS and other representations in F1 score for low Decision Tree depths in any of these domains, with the exception of the depth-two trees for sentiment. Despite MDS not having informative dimensions, our method is still able to obtain a distangled feature representation from this document embedding. There is a low correlation between performance on the raw dimensions of the space and performance with rankings on directions in low-depth Decision Trees. Despite the information encoded in the space, if it is not disentangled then the classifier will not perform well.

### 4.8.5 Word Directions

In Table ?? the results for the feature representation composed of rankings of entities on single-term directions that performed best when used as input to a classifier is shown.

As can be seen for the newsgroups results in Table ??, the number of dimensions now has a bigger impact on the results, in particular for the depth-three and two trees. This difference was

Newsgroups	D1			D2			D3			DN			SVM		
	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Rec
PCA 200	0.701	0.251	0.148	0.811	0.843	0.366	0.245	0.719	0.956	0.355	0.54	0.265	0.946	0.44	0.45
PCA 100	0.698	0.247	0.146	0.813	0.835	0.362	0.241	0.731	0.957	0.356	0.576	0.257	0.948	0.451	0.465
PCA 50	0.68	0.24	0.141	0.829	0.834	0.355	0.234	0.735	0.957	0.329	0.472	0.253	0.947	0.45	0.462
AWV 200	0.687	0.217	0.126	0.781	0.758	0.256	0.156	0.718	0.764	0.26	0.157	0.751	0.937	0.339	0.352
AWV 100	0.677	0.21	0.122	0.775	0.78	0.275	0.173	0.683	0.746	0.25	0.149	0.769	0.934	0.324	0.332
AWV 50	0.696	0.219	0.127	0.772	0.777	0.272	0.168	0.71	0.743	0.25	0.149	0.786	0.935	0.325	0.335
MDS 200	0.581	0.184	0.103	<b>0.837</b>	0.742	0.262	0.16	0.729	0.719	0.236	0.139	0.785	0.935	0.327	0.332
MDS 100	0.586	0.187	0.105	0.833	0.754	0.261	0.159	0.727	0.705	0.236	0.138	<b>0.808</b>	0.935	0.33	0.338
MDS 50	0.593	0.153	0.087	0.647	0.716	0.25	0.15	<b>0.756</b>	0.736	0.243	0.144	0.774	0.935	0.324	0.335
D2V 200	0.682	0.205	0.119	0.746	0.802	0.268	0.169	0.646	0.77	0.269	0.164	0.75	0.94	0.366	0.389
D2V 100	0.682	0.208	0.12	0.762	0.792	0.268	0.168	0.662	0.786	0.268	0.164	0.727	0.94	0.376	0.392
D2V 50	0.683	0.207	0.12	0.764	0.809	0.294	0.187	0.694	0.782	0.28	0.172	0.761	0.943	0.394	0.415
PPMI	<b>0.948</b>	0.33	<b>0.532</b>	0.239	0.947	0.407	0.511	0.338	0.944	<b>0.444</b>	0.506	0.396	<b>0.951</b>	<b>0.494</b>	<b>0.496</b>
Topic	0.852	<b>0.431</b>	0.304	0.743	<b>0.96</b>	<b>0.423</b>	<b>0.604</b>	0.326	<b>0.961</b>	0.444	<b>0.606</b>	0.35	0.944	0.432	0.434
													0.879	0.46	0.318
															<b>0.835</b>

Table 4.7: Best results for the newsgroups for a variety of classifiers when using the document embeddings as input.

Reuters	D1		D2		D3		DN		SVM		Sentiment		D1		D2		D3		DN		SVM	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
PCA	0.847	0.328	0.917	0.413	0.978	0.501	0.978	0.565	0.989	0.761	PCA	0.745	0.705	0.755	0.77	0.778	0.773	<b>0.781</b>	<b>0.779</b>	<b>0.891</b>	<b>0.893</b>	
AWV	0.782	0.252	0.971	0.328	0.974	0.417	0.973	0.495	0.987	0.719	AWV	0.642	0.652	0.643	0.694	0.695	0.717	0.66	0.663	0.827	0.829	
MDS	0.791	0.263	0.9	0.357	0.979	0.489	0.976	0.522	0.988	0.67	D2V	0.642	0.664	0.66	0.707	0.702	0.7	0.711	0.708	0.878	0.878	
D2V	0.818	0.268	0.867	0.298	0.974	0.445	0.971	0.482	0.986	0.724	PPMI	0.616	0.7	0.655	0.719	0.675	0.73	0.712	0.71	0.887	0.888	
PPMI	<b>0.975</b>	<b>0.616</b>	<b>0.978</b>	<b>0.699</b>	<b>0.98</b>	<b>0.723</b>	<b>0.984</b>	<b>0.746</b>	<b>0.99</b>	<b>0.8</b>	Topic	<b>0.793</b>	<b>0.79</b>	<b>0.794</b>	<b>0.791</b>	<b>0.81</b>	<b>0.811</b>	0.733	0.73	0.815	0.822	
Topic	0.92	0.411	0.977	0.527	0.977	0.536	0.977	0.56	0.95	0.513												
Placetypes	D1		D2		D3		DN		SVM		Movies	D1		D2		D3		DN		SVM		
OpenCYC	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	Genres	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	
PCA	0.586	0.346	0.708	0.343	0.695	0.342	0.832	0.309	0.847	0.474	PCA	0.722	0.301	0.755	0.339	0.717	0.321	0.884	0.372	<b>0.925</b>	0.518	
AWV	0.625	<b>0.383</b>	0.651	0.376	0.728	<b>0.396</b>	<b>0.844</b>	<b>0.362</b>	0.85	0.466	AWV	0.679	0.29	0.774	0.321	0.756	0.343	0.873	0.312	0.922	0.496	
MDS	0.624	0.364	0.7	<b>0.397</b>	0.731	0.374	0.843	0.305	0.861	<b>0.476</b>	MDS	0.679	0.298	0.79	0.358	0.773	0.354	0.887	0.385	0.875	<b>0.532</b>	
PPMI	<b>0.728</b>	0.371	0.75	0.351	0.739	0.352	0.843	0.323	<b>0.9</b>	0.366	PPMI	<b>0.852</b>	<b>0.429</b>	<b>0.91</b>	0.443	<b>0.912</b>	<b>0.483</b>	0.882	<b>0.416</b>	0.923	0.526	
Topic	0.708	0.365	<b>0.87</b>	0.271	<b>0.87</b>	0.313	0.831	0.313	0.808	0.407	Topic	0.767	0.415	0.905	<b>0.472</b>	0.912	0.455	<b>0.889</b>	0.415	0.843	0.491	
Placetypes	D1		D2		D3		DN		SVM		Movies	D1		D2		D3		DN		SVM		
Foursquare	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	Keywords	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	
PCA	0.731	0.342	0.823	0.393	0.86	0.388	0.887	0.398	0.896	0.568	PCA	0.647	0.185	0.644	0.193	0.677	0.199	0.846	0.161	0.787	0.272	
AWV	0.767	0.401	0.828	0.478	0.85	0.452	0.905	<b>0.505</b>	0.923	<b>0.622</b>	AWV	0.5	0.16	0.641	0.179	0.595	0.174	0.853	0.141	0.717	0.23	
MDS	<b>0.915</b>	0.438	0.804	0.427	0.86	0.454	0.893	0.462	0.932	0.619	MDS	0.633	0.179	0.69	0.198	0.674	0.201	0.84	0.163	0.788	<b>0.28</b>	
PPMI	0.889	0.473	0.915	<b>0.512</b>	0.904	0.491	0.881	0.31	<b>0.938</b>	0.567	PPMI	<b>0.818</b>	<b>0.243</b>	0.745	<b>0.224</b>	0.739	<b>0.224</b>	0.847	<b>0.17</b>	<b>0.921</b>	0.217	
Topic	0.864	<b>0.488</b>	<b>0.916</b>	0.433	<b>0.917</b>	<b>0.526</b>	<b>0.907</b>	0.464	0.916	0.569	Topic	0.629	0.189	<b>0.932</b>	0.05	<b>0.93</b>	0.075	<b>0.857</b>	0.152	0.678	0.21	
Placetypes	D1		D2		D3		DN		SVM		Movies	D1		D2		D3		DN		SVM		
Geonames	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	Ratings	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	
PCA	0.502	0.301	0.69	0.305	0.68	0.295	0.821	0.243	0.844	0.401	PCA	<b>0.65</b>	0.463	0.681	<b>0.475</b>	0.684	<b>0.486</b>	0.744	0.408	0.771	0.58	
AWV	0.657	0.326	0.755	0.323	0.842	<b>0.367</b>	0.813	0.332	0.865	<b>0.514</b>	AWV	0.601	0.423	0.618	0.433	0.596	0.448	0.736	0.372	0.73	0.532	
MDS	0.626	0.349	0.695	<b>0.34</b>	0.796	0.272	<b>0.845</b>	0.295	0.638	0.397	MDS	0.592	0.437	0.635	0.449	0.631	0.452	<b>0.752</b>	<b>0.412</b>	0.773	<b>0.589</b>	
PPMI	<b>0.808</b>	0.361	0.732	0.301	0.76	0.242	0.83	0.283	<b>0.894</b>	0.312	PPMI	0.583	0.47	0.635	0.453	0.605	0.453	0.73	0.384	<b>0.825</b>	0.536	
Topic	0.771	<b>0.365</b>	<b>0.863</b>	0.3	<b>0.85</b>	0.219	0.828	<b>0.348</b>	0.819	0.349	Topic	0.575	<b>0.473</b>	<b>0.789</b>	0.243	<b>0.789</b>	0.38	0.739	0.375	0.704	0.501	

Table 4.8: Best results for all domains for a variety of classifiers when using the document embeddings as input.

smaller for the depth-one trees. This is likely because despite the size of the document embedding model, they all modelled a similar most-important single direction relevant to the task, e.g. a direction that can classify if the entity is religious like the word-direction for "celestial". However, variants of the space size change the representation of auxiliary directions that are not as relevant to the task but are useful when the depth is higher for the tree.

The best space type also varied across domains, e.g. the classifiers performed best for reuters with a disentangled single-term feature representation obtained from MDS used as input, but a feature representation obtained from Doc2Vec when used as input performed best for newsgroups. Loosely, it is possible to attribute the performance increase for a space-type to an increased score (e.g. NDCG score) for relevant directions to the class. When looking at the qualitative results, generally the words common to all space-types are the most salient, as can be seen in Table 4.3, so in this case the increased results for different hyper-parameters likely comes down to either these common directions being better modelled in the representation or terms that were not modelled well before that were relevant to the task being modelled better.

We see that generally, the best document embedding model is the same across the same domain: specifically AWV is the best for the place-types but MDS is best for the movies (despite a marginal difference in the ratings). Generally, this means that these document embedding types in particular are good at modelling informative properties for the tasks.

Although some tasks on some classifiers did perform better using score-types like F1-score, in general, filtering directions by NDCG score performed the best. In particular it was found the best score-type for Sentiment, Newsgroups, Reuters, Movies Genres, Movies Keywords in depth-3 Decision Trees. In conclusion, the document embedding model is an important hyper-parameter to tune for each domain, as is the size of that document embedding. However, document embeddings that perform well on one task may be good at the others too. Finally, NDCG is a good scoring metric to use for finding features that are meaningful in the representation.

#### 4.8.6 Clustered Directions

??

Clustering has two main goals: The first is that features that align with properties of entities in

Newsgroups	D1				D2				D3			
	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec
PCA 200	0.955	0.348	0.521	0.261	0.959	0.424	0.678	0.309	0.96	0.454	0.674	0.343
PCA 100	0.957	0.382	0.491	0.313	0.961	0.474	0.679	0.364	<b>0.963</b>	0.512	0.694	0.406
PCA 50	0.957	0.373	0.417	0.337	<b>0.963</b>	0.478	0.621	0.388	0.963	0.506	0.7	0.396
AWV 200	0.832	0.35	0.226	0.777	0.957	0.383	0.517	0.305	0.958	0.445	0.598	0.354
AWV 100	0.83	0.343	0.219	0.785	0.823	0.36	0.233	<b>0.792</b>	0.956	0.387	0.563	0.295
AWV 50	0.807	0.341	0.215	0.816	0.833	0.361	0.236	0.762	0.954	0.392	0.511	0.318
MDS 200	<b>0.959</b>	<b>0.418</b>	<b>0.543</b>	0.339	0.962	0.465	0.669	0.357	0.962	0.493	<b>0.707</b>	0.379
MDS 100	0.857	0.365	0.244	0.725	0.959	0.428	0.624	0.326	0.96	0.453	0.644	0.349
MDS 50	0.821	0.324	0.206	0.762	0.842	0.386	0.258	0.77	0.957	0.398	0.596	0.299
D2V 200	0.831	0.343	0.22	0.784	0.96	0.47	<b>0.683</b>	0.358	0.962	0.494	0.69	0.385
D2V 100	0.844	0.374	0.243	0.803	0.961	<b>0.49</b>	0.642	0.396	0.962	0.517	0.67	0.421
D2V 50	0.845	0.388	0.252	<b>0.844</b>	0.962	0.488	0.639	0.395	0.963	<b>0.537</b>	0.673	<b>0.446</b>
Reuters												
	D1	D2	D3	Sentiment	D1	D2	D3		D1	D2	D3	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
PCA	0.976	0.658	0.979	0.679	0.977	0.467	PCA	0.739	0.759	<b>0.797</b>	<b>0.814</b>	0.802
AWV	0.975	0.598	0.979	0.656	0.98	0.66	AWV	0.7	0.699	0.711	0.736	0.735
MDS	0.975	<b>0.678</b>	<b>0.98</b>	<b>0.706</b>	<b>0.982</b>	<b>0.72</b>	D2V	<b>0.776</b>	<b>0.784</b>	0.782	0.801	<b>0.822</b>
D2V	<b>0.977</b>	0.583	0.979	0.664	0.98	0.632						<b>0.821</b>
Placetypes												
	D1	D2	D3	Movies	D1	D2	D3		D1	D2	D3	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
OpenCYC	0.632	0.371	0.704	0.381	0.735	0.365	PCA	0.824	0.412	0.82	0.441	0.913
PCA	<b>0.66</b>	<b>0.404</b>	<b>0.734</b>	<b>0.428</b>	<b>0.755</b>	<b>0.39</b>	AWV	0.81	0.421	0.837	0.436	0.912
AWV	0.658	0.374	0.711	0.385	0.746	0.35	MDS	<b>0.849</b>	<b>0.446</b>	<b>0.839</b>	<b>0.463</b>	<b>0.918</b>
MDS	0.658	0.374	0.711	0.385	0.746	0.35						<b>0.495</b>
Foursquare												
	ACC	F1	ACC	F1	ACC	F1	Keywords	ACC	F1	ACC	F1	ACC
PCA	0.785	0.477	<b>0.907</b>	0.474	0.869	<b>0.531</b>	PCA	0.737	0.225	0.727	0.227	<b>0.709</b>
AWV	<b>0.918</b>	<b>0.541</b>	0.881	<b>0.498</b>	0.889	0.466	AWV	0.656	0.201	0.672	0.203	0.652
MDS	0.82	0.416	0.879	0.482	<b>0.897</b>	0.485	MDS	<b>0.745</b>	<b>0.23</b>	<b>0.74</b>	<b>0.233</b>	0.708
Geonames												
	ACC	F1	ACC	F1	ACC	F1	Ratings	ACC	F1	ACC	F1	ACC
PCA	0.665	0.348	0.754	0.342	0.743	0.306	PCA	<b>0.647</b>	<b>0.466</b>	<b>0.721</b>	<b>0.499</b>	0.681
AWV	<b>0.711</b>	<b>0.444</b>	<b>0.795</b>	<b>0.533</b>	<b>0.802</b>	<b>0.473</b>	AWV	0.646	0.463	0.692	0.474	0.677
MDS	0.591	0.289	0.772	0.333	0.764	0.352	MDS	0.62	0.463	0.692	0.489	<b>0.686</b>

Table 4.9: The best results (hyper-parameter tuned) for the classifiers for each task when the disentangled feature representations composed of rankings of entities on single-term directions..



the domain are obtained by combining together single-word directions that reference phenomena, e.g. in a domain of movie reviews "Blood", "Scary", "Horror" can be clustered to obtain a cluster that better models the idea of horror in film, rather than modelling individual things that occur in film. The second is that it makes the features more interpretable by providing context to the words. In this section, we examine how these more abstract and complex cluster features perform quantitatively at key domain tasks.

K-means mostly outperforms Derrac. It does not in the case of Keywords, where Derrac performs better for every Decision Tree. Although the differences in absolute values are quite small in this case, it is still significant as it is quite difficult to achieve high performance on this task, making these relative changes important. This case can give us insight into how disentanglement affects performance on different classes and domains - and how our unsupervised method selects the best parameters. Specifically, when looking into the how the individual classes fared, when using 100 clusters, the Derrac variant clusters performed better at the keywords "shot-in-the-chest" and "machine-gun" and sacrificed performance in the "sequel" class. In Derrac, there was the following cluster ("soldiers combat fighting military battle ... weapons rambo gunfights spaghetti guns ...") while for the results of the k-means clustering method when restricted to 200 clusters these words were split into two separate clusters, one for guns ("gun explosions shoot shooting weapons ... rambo") and one for military ("war soldiers combat military ... platoon infantry"). It is possible that as the Derrac method combined these together into a single cluster it was able to better capture the classes for "shot-in-the-chest" and "machine-guns" as these keywords refer to war films where people were shot or shooting. So in this case, the clusters chosen by the Derrac variant supported the classification of the documents.

This idea is supported when looking at the depth-three tree for this class, which uses this cluster as its first node as well as a node in the depth-two layer. This is an instance where averaging the direction of a heavily populated cluster performs better than obtaining features that are more dense and less disentangled.

Meanwhile, this same lack of separation caused the clusters to lose performance in the "sequel" class. With the K-means method, the cluster was found for ("franchise sequels sequel installments") with the Derrac variant, the corresponding cluster was ("franchise sequels sequel instalments entry returns"). This cluster was also chosen for the Derrac variant as the first node of its Decision Tree, but this caused it to perform worse than k-means. This is likely because

although the words "entry" and "returns" were most similar to this cluster, they disrupted the direction too much. Indeed, when looking at the k-means clusters, the "returns" direction is clustered with "events situation conclusion spoiler ... protagonists exscapes break scenario ...", seemingly referring to a character or thing "returning" in a conclusive part of the movie, and the word "entry" is clustered with the words "effective genuine ... hits build surprisingly ... succeeds essentially finale entry ..." seemingly relating to a more sentiment related cluster about how a movie performed. So in this case k-means being able to find more disentangled clusters than Derrac gave it a performance advantage.

This could be due to the best-performing variants of the Derrac clustering method obtaining 100 clusters (meaning the clusters would contain more terms) and the best performing variants of the k-means obtaining 200 clusters. However, in the 100-size k-means clusters, "gun" and "explosions" ended up being in a cluster with ("western outlaw heist shootout west"), making it a more western oriented cluster, and the idea of a war was even more separated with a single cluster corresponding to ("war soldiers military soldier army sergeant sgt platoon infantry"). In conclusion, Derrac for the Keywords task captured certain properties better than k-means, in particular by clustering together the idea of "war" and "guns" to achieve high performance on the keywords "shot-in-the-chest" and "machine-guns". k-means favoured a more separated approach to these ideas, which meant that although it captured the idea of "war" well, it was not able to capture the classes inbetween the idea of "war" and "guns".

#### 4.8.7 Conclusion

In conclusion, this chapter introduced a methodology to go from a Vector Space Model of Semantics and an associated bag-of-words to a disentangled representation, and these disentangled representations are shown to be disentangled by performing well on simple classifiers. The disentangled representation in this work has natural labels, and has the potential to be applied in a simple interpretable classifier e.g. a decision tree as its components correspond to properties. This methodology could be used as an alternative to methods like Topic Models, and give insight into the parameters required and qualitative results that can be obtained. This method can be applied to many vector spaces and domains as a post-processing step, and is extensively tested the qualitatively and quantitatively, in particular qualitatively finding that the variants

which perform best in the quantitative evaluation also make sense from a qualitative point of view tend to use properties that make good intuitive sense. We find that our method greatly outperforms the original representations on low-depth Decision Trees, giving good evidence that the representation is disentangled, strongly suggesting that the features that are obtained with this method are indeed semantically meaningful. Additionally, the results of these simple classifiers are competitive with standard interpretable representation baselines in most cases. Variations are introduced to work by [?] and these variants are found to generally perform well, in particular the NDCG scoring method performs better than Kappa in most cases, and that standard K-means performs better than the original clustering method. A variety of space-types and domains were experimented with, verifying that the methodology can be applied more generally than shown in [?]. The main experiments that would be interesting to expand on for this chapter would be more state-of-the-art representations, specific investigations of how those representations are able to achieve such strong results, and interpretability experiments to see how the cluster labels fare in real-world situations.

Newsgroups	D1			D2			D3					
	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec
K-means 200	<b>0.852</b>	<b>0.394</b>	<b>0.261</b>	0.795	<b>0.958</b>	<b>0.433</b>	<b>0.58</b>	0.345	<b>0.963</b>	<b>0.513</b>	<b>0.704</b>	0.403
K-means 100	0.842	0.388	0.257	0.791	0.958	0.366	0.516	0.284	0.962	0.5	0.635	<b>0.412</b>
K-means 50	0.834	0.381	0.248	<b>0.819</b>	0.815	0.336	0.212	<b>0.81</b>	0.961	0.485	0.612	0.402
Derrac 200	0.803	0.313	0.202	0.693	0.797	0.306	0.191	0.781	0.958	0.409	0.605	0.309
Derrac 100	0.792	0.305	0.197	0.667	0.791	0.287	0.179	0.721	0.957	0.374	0.56	0.281
Derrac 50	0.769	0.26	0.162	0.661	0.768	0.237	0.143	0.693	0.955	0.315	0.47	0.237

Table 4.10: All clustering size results for the newsgroups

Reuters	D1		D2		D3		Sentiment		D1		D2		D3	
	ACC	F1	ACC	F1	ACC	F1	F1	K-means	ACC	F1	ACC	F1	ACC	F1
K-means	<b>0.875</b>	<b>0.338</b>	<b>0.975</b>	<b>0.54</b>	0.973	<b>0.58</b>	<b>0.58</b>	K-means	0.623	0.674	<b>0.837</b>	<b>0.844</b>	0.658	0.707
Derrac	0.797	0.291	0.973	0.402	<b>0.974</b>	0.485	Derrac	<b>0.712</b>	<b>0.735</b>	0.802	0.82	<b>0.803</b>	<b>0.813</b>	
Placetypes	D1		D2		D3		Movies	D1		D2		D3		
OpenCYC	ACC	F1	ACC	F1	ACC	F1	Genres	ACC	F1	ACC	F1	ACC	F1	
K-means	<b>0.641</b>	<b>0.413</b>	<b>0.735</b>	<b>0.405</b>	0.75	<b>0.43</b>	K-means	<b>0.813</b>	<b>0.431</b>	<b>0.913</b>	<b>0.513</b>	<b>0.913</b>	<b>0.506</b>	
Derrac	0.605	0.39	0.672	0.392	<b>0.755</b>	0.391	Derrac	0.759	0.341	0.789	0.431	0.911	0.432	
Foursquare	ACC	F1	ACC	F1	ACC	F1	Keywords	ACC	F1	ACC	F1	ACC	F1	
K-means	<b>0.913</b>	<b>0.462</b>	<b>0.911</b>	<b>0.5</b>	<b>0.891</b>	<b>0.511</b>	K-means	0.667	0.208	0.648	0.202	0.678	0.213	
Derrac	0.768	0.392	0.835	0.445	0.805	0.425	Derrac	<b>0.726</b>	<b>0.215</b>	<b>0.745</b>	<b>0.22</b>	<b>0.707</b>	<b>0.219</b>	
Geonames	ACC	F1	ACC	F1	ACC	F1	Ratings	ACC	F1	ACC	F1	ACC	F1	
K-means	<b>0.772</b>	0.43	<b>0.774</b>	0.407	<b>0.819</b>	<b>0.472</b>	K-means	<b>0.671</b>	<b>0.504</b>	0.638	<b>0.507</b>	<b>0.686</b>	<b>0.513</b>	
Derrac	0.678	<b>0.449</b>	0.74	<b>0.411</b>	0.807	0.415	Derrac	0.651	0.445	<b>0.669</b>	0.463	0.627	0.479	

**Table 4.11: The best results when using the entities ranked on the directions output by the clustering method for the optimal number of clusters for each domain and task..**

# Directions in Neural Networks

## 5.1 Introduction

The previous chapter showed how in a vector space of entities (e.g. different movies) fine-grained semantic relationships can be identified with directions (e.g. more violent than) and used as features in a document classification task. This Chapter explores how these fine-grained semantic relationships can be used in relationship to neural networks. Two kinds of neural networks are investigated, feed-forward networks and auto-encoders. Vector space embeddings in the hidden layers of feedforward networks are re-organized into interpretable feature representations, and they are qualitatively investigated using examples and low-depth decision trees. Interpretable feature-representations are obtained from the layers of auto-encoders, and their use is investigated in the context of learning about relationships between properties in the domain. Specifically, auto-encoders are used to obtain a sequence of increasingly small entity embeddings that model increasingly abstract relationships. After identifying directions that model important properties of entities in each of these vector spaces, symbolic rules are induced that relate specific properties to more general ones. Illustrative examples are provided of the hidden layers and these rules.

The approach in this Chapter builds on the method from Chapter 6, finding directions which correspond to interpretable properties in neural network vector spaces, labelled using adjectives and nouns that appear in the text collection. In order to obtain the symbolic rules, first a series of increasingly general entity embeddings is derived using auto-encoders (see Section 5.3.1). To induce rules from embeddings, properties derived from those embeddings are linked together. As an example, below is one of the rules derived with this method, where the first two terms are

from one entity embedding, and the final term is from a more abstract entity embedding.

$$\text{IF Emotions AND Journey THEN Adventure} \quad (5.1)$$

Generally, there are two goals to achieve for this method for the auto-encoders: First, qualitatively investigate if neural networks can help identify relationships between properties. Second, to investigate the kind of properties encoded in auto-encoders. One potential application of this work in recommendations is in the robustness of explanations using these properties. In the domain of movies, we may have a situation where the synopsis or reviews mention the words "Emotions" and "Journey", from which the system could derive that it is probably an "Adventure" movie and use that term as a part of its supporting explanation e.g. "You liked how this movie was an emotional and a journey, so you may like this adventure movie." Of-course if these properties are indeed salient and can be linked in this way, these properties are also be useful for investigating how deep neural networks represent information.

Additionally, Feed-forward networks are qualitatively investigated using properties and associated entities ranked on those properties obtained by using the hidden-layer as the starting vector space representation. It is found that interpretable features can indeed be obtained from these neural network representations, and that the rankings of entities on these properties make sense. Additionally, in the case of the newsgroups the predictive performance of a Depth-3 Decision tree that uses the cluster-features as input can outperform the original neural network, and in general the performance of the neural network is retained or slightly decreased in comparison to depth-3 decision trees learned using properties derived from their hidden layers.

In conclusion, feed-forward networks and auto-encoders are investigated using the methods described in Chapter 6. In Section some neural-network specific methods for interpretability are discussed. Following that in Section 5.3, the detail for how we obtain the hidden layer of the feed-forward networks and how the method from Chapter 6 is used to investigate stacked denoising auto-encoders, and how rules are induced that explain the semantic relationships between the properties that are discovered. Next in Section ?? the results of the qualitative and quantitative investigation of the feed-forward networks is discussed followed by the qualitative results for the auto-encoders. Finally in ?? conclusions are drawn as to the usefulness of the method and its limitations. In particular, we find that the properties are meaningful for neural networks and provide some insight but it would be useful to improve the properties such that they retain

interpretability even when being made abstract by denoising auto-encoders, which is what is covered in Chapter ??.

## 5.2 Background

There are many methods for obtaining an explanation of a black-box after it has been learned using a 'proxy model', a model that approximates the decisions made by the network but is interpretable. There are three general approaches: LIME that produces a linear model that explains the network in terms of input features [?], DeepRed that creates a decision tree that is faithful to the original network [?], and automatic rule extraction [?]. LIME differs from the work in this thesis as it focuses on the relationship between input and output, rather than explaining the internal layers of the network. Essentially, the work on this thesis focuses on layers, while their work focuses on explaining a black-box in terms of its input.

Although the decision tree method seems similar on the surface, it differs from the work here as the decision tree in this work is obtained from meaningful interpretable features from the re-organized neural network space, rather than attempting to produce a proxy model decision tree that is faithful to the neural network. Essentially, the work in this chapter is investigating the re-organization of the neural network into interpretable features, and a decision tree is a simple interpretable model that can help us investigate those interpretable features. A variety of simple interpretable models could be used.

The existing neural network rule extraction algorithms can be categorized as either decompositional, pedagogical or eclectic [?]. Decompositional approaches derive rules by analysing the units of the network, while pedagogical approaches treat the network as a black box, and examine the global relationships between inputs and outputs. Eclectic approaches use elements of both decompositional and pedagogical approaches. Our method could be classified as decompositional, as we re-organize the hidden layer of the neural network. We will now describe some similar approaches and explain how our methods differs.

The algorithm in [?] is a decompositional approach that applies to a neural network with two hidden layers. It uses hyperplanes based on the weight parameters of the first layer, and then combines them into a decision tree. NeuroLinear [?] is a decompositional approach applied to



a neural network with a single hidden layer that discretizes hidden unit activation values and uses a hyperplane rule to represent the relationship between the discretized values and the first layer's weights. HYPINV [?] is a pedagogical approach that calculates changes to the input of the network to find hyperplane rules that explain how the network functions.

The main difference in our work is that our method induces rules from properties derived from the re-organized layers of a network, rather than learning rules that describe the relationships between units in the network itself. Additionally, the focus is on a qualitative investigation of the network and its potential to learn increasingly general entity embeddings from hidden representations rather than tuning network parameters such that weights directly relate to good rules.

Another recent topic that relates to our work is improving neural networks and entity embeddings using symbolic rules [?]. In [?] a combination of first-order logic formulae and matrix factorization is used to capture semantic relationships between concepts that were not in the original text. This results in relations that are able to generalize well from input data.

This is essentially the opposite of the task we consider in this paper: using embeddings to learn better rules. The rules that we derive are not intended to explain how the network functions but rather to describe the semantic relationships that hold in the considered domain. In other words, our aim is to investigate the use of neural network representations in the hidden layer as a tool for learning logical domain theories, where the focus is on producing rules that capture meaningful semantic relationships.

## 5.3 Method

A bag-of-words representation when used as input to a decision-tree classifies using many individual words, for example the class of "comedy" would have a node for each feature, and those features would be words like "laughed" "witty" "charming" and so on. Because of this, a larger tree is required to classify well.

Neural networks with large hidden layers can also be expected to have similar behaviour. If the hidden layer is larger, then the representation will contain directions that correspond to more granular concepts. For example, to classify comedy there may be directions like "gags,

slapstick, gag", "witty, charming, wit", "comedies, comedy" "eccentric", simple concepts that directly relate to the class.

As the layers get smaller, the concepts become more condensed and general. To see an example of this, see 5.4.3 where example clustered features from the best performing decision trees are shown. The method in this section for the feedforward network is essentially to use a reasonably sized hidden layer that will result in good properties, and then take that hidden layer representation and investigate it qualitatively. In the case of the auto-encoders, instead successively smaller, more abstract, less disentangled representations are obtained and they and the relationships between them are investigated.

### 5.3.1 Inducing Rules from Auto-Encoder Entity Embeddings

In this section, we explain how we obtain a series of increasingly general entity embeddings, and how we can learn symbolic rules that link properties from subsequent spaces together.

To construct more general embeddings from the initial embedding provided by the MDS method, we use stacked denoising auto-encoders [?]. Standard auto-encoders are composed of an “encoder” that maps the input representation into a hidden layer, and a “decoder” that aims to recreate the input from the hidden layer. Auto-encoders are normally trained using an objective function that minimizes information loss (e.g. Mean Squared Error) between the input and output layer [?]. The task of recreating the input is made non-trivial by constraining the size of the hidden layer to be smaller than the input layer, forcing the information to be represented using fewer dimensions, or in denoising auto-encoders by corrupting the input with random noise, forcing the auto-encoder to use more general commonalities between the input features. By repeatedly using the hidden layer as input to another auto-encoder, we can obtain increasingly general representations. To obtain the entity representations from our auto-encoders, we use the activations of the neurons in a hidden layer as the coordinates of entities in a new vector space.

The main novelty of our approach is that we characterize the salient properties (i.e. clusters of directions) modelled in one space in terms of salient properties that are modelled in another space. Specifically, we use the off-the-shelf rule learner JRip [7] to predict which entities will be highly ranked, according to a given cluster direction, using as features the rankings induced by the clusters of the preceding space. To improve the readability of the resulting rules, rather

than using the precise ranks as input, we aggregate the ranks by percentile, i.e 1%, 2%, ..., 100%, where an entity has a 1% label if it is among the 1% highest ranked entities, for a given cluster direction. For the class labels, we define a movie as a positive instance if it is among the highest ranked entities (e.g. top 2%) of the considered cluster direction. Using the input features of each layer and the class labels from the subsequent layer, these rules can be used to explain the semantic relationships between properties modelled by different vector spaces. We note that one drawback of discretizing continuous attributes is that the accuracy of the rules extracted from the network may decrease [?]. However, in our setting, interpretability is more important than accuracy, as we do not aim to use these rules for making predictions, but use them only for generating explanations and getting insight into data.

### 5.3.2 Feed-forward networks

We train two neural networks with non-linear activation functions. The first is NNET-U, a neural network with a single hidden layer of a size learned using hyper-parameter optimization that uses an already learned unsupervised representation as input. The second is NNET-B, a neural network that starts with the PPMI BOW as input. The output layer is the of size  $C_n$  where  $C_n$  is the number of classes. Each network is trained on one task from each domain. As the goal is a qualitative analysis, these tasks were chosen as it is clear what properties will correspond to them. To obtain a representation from this neural network, we simply take the entities as they are represented by the trained model in the hidden layer.

The representations these networks build should differ from those obtained in the unsupervised representation in a few different ways. First, the neural network used is multi-label, meaning that all of the classification objectives are trained at the same time. Second, a non-linear activation function is used, sometimes in multiple layers. Finally, the supervised objective should shape the space more than an unsupervised objective, although only objectives that are relevant to the domain are considered in this case - as we are not interested in a network that ignores a large amount of information, rather we want the network to construct a representation that more accurately represents domain knowledge.

Following results showing that simple neural networks can still perform well on text classification without needing a complex architecture [?] [?], we use a neural network with the following

parameters: Cross-entropy loss, Adagrad trainer with default parameters, Dropout, Sigmoid activation on the output layer. Instead of Relu which we found did not obtain good directions, we used the tanh activation function. Dropout is a regularizer that drops out units during training and makes the representation more general and robust. Adagrad is a trainer that adjusts the learning rate during training.

We additionally learn a neural network starting from the bag-of-words with two hidden-layers, the first of size 1000 and the second of size 100. In preliminary experiments, the network with two layers outperformed the single layer network consistently, and in the case of newsgroups performed more strongly than an SVM on the unsupervised representation. When taking a representation from this neural network, we take this final hidden layer of size 100. This is the representation used in the qualitative examples.

## 5.4 Investigating Feedforward Neural Networks

### 5.4.1 Parameters

We use three different domains: Newsgroups, Placetypes and Movie reviews. We choose newsgroups and movie reviews as they have a relevant task that can be used to verify if the network has learned general domain concepts, e.g. in the case of newsgroups the natural categories of the documents and the movie reviews the genres of movies. Placetypes are chosen not because we expect them to perform well with neural networks, but rather because their entities are easy to understand and in-turn explain. In fact, as placetypes has such a low number of entities for its tasks we cannot expect good results with neural networks which typically perform well with a large number of entities.

When obtaining the directions, we set the frequency cut-off to the top 10,000 words and the direction cut-off when classifying to 2000 features. This is arbitrary, as both of these cut-offs need to be tuned for the specific space-type and task in-order to achieve strong results. The reason that these parameters are not tuned in this case is because we are interested in the qualitative nature of the directions, not the performance on the text classification task. Additionally, as Normalized Discounted Cumulative Gain (See Section 4.5.2) was shown to perform well for a variety of embedding types and domains, it was the only scoring method used.

To learn a baseline feedforward network, the highest performing representation found in Table 4.8.4 for cluster-features is used as input to depth-3 decision trees and use those representations as input to the neural network. For the newsgroups, this is the size 100 Doc2Vec space. Then, we set the hidden-layer to be the same size as the input representation and use the "tanh" activation function. The reasoning behind this is twofold: First, we can assume that the representation that performs well on depth-three decision trees contain good interpretable concepts that can be meaningfully adjusted by the network, and second that this allows for easy comparison between the original representation and the representation taken from the hidden layer of the network, as the highest performing directions can be compared.

The network is hyper-parameter tuned with the following values:

- $epoch = [100, 200, 300]$
- $dropout = [0.1, 0.25, 0.5, 0.75]$
- $hidden_{layer\_size} = [1, 2, 3, 4]$

The batch size is 100 for all experiments, excluding the place-types which used a batch-size of 10 as there were so few entities. When using the bag-of-words as input to the neural network we followed "default" parameters as recommended by [?] with an initial hidden layer of size 1000. However, a secondary hidden layer of size 100 is added for this work, chosen as it is close to the size of the original unsupervised representations.

For the linear SVM and the Decision Trees of depth-3, hyper-parameter optimization for these models was used with the same parameters as described in Section 4.8.2.

## 5.4.2 Quantitative Results for Feedforward Network

The results for place-types were obtained as their entities can be easily understood without expert knowledge (e.g. "tree" and "cliff" rather than "The Shining" in the case of the movies). It is not unexpected that they performed poorly in the neural networks, as they have a very low number of entities (391) and it is also an unbalanced task.

Here, we show quantitative results for:

	Movies	Placetypes	Newsgroups
Unsupervised (Linear SVM)	0.532082959	0.630040432	0.628171051
NNET-U	0.559090895	0.563313632	0.627577055
NNET-B	0.435345945	0.597008771	0.673618458
Directions Unsupervised DT3	0.493072458	0.508163669	0.536686468
Directions NNET-U DT3	0.505155338	0.460648739	0.50988517
Directions NNET-B DT3	0.501545907	0.521526974	0.664793407
Clusters Unsupervised DT3	0.506096231	0.544001448	0.513367341
Clusters NNET-U DT3	0.472297312	0.541887565	0.478077988
Clusters NNET-B DT3	0.501510396	0.5967867	0.682928611

**Table 5.1: F1-scores for each embedding type and domain**

- The best performing linear SVM result on an unsupervised representation from Chapter 6.
- The predictions of the neural networks.
- The results for the directions obtained from the hidden layers of the neural networks on depth-3 decision trees.
- The results of the clustering of those directions on depth-3 decision trees.

The intention of the results in this section is not to achieve state-of-the-art on the task, rather it is to show that depth-3 decision trees that use the directions obtained from a simple neural network perform reasonably well on the task or sometimes better. As in Chapter 6, this validates that these features are indeed important for the domain. The idea that interpretable features can be obtained from the neural networks using the method in Chapter 6 is validated and not much predictive power is lost over the original neural network in some cases. As these features are also able to perform well in the simple classifier, it validates the use of Decision Trees in the qualitative investigation in Section 5.4.3.

These results are also preliminary insights that inform what we expect the directions to be like in in Section 5.4.3. In the case of the movies we know that the neural network with the unsupervised embedding as input (NNET-U) performed better than the linear SVM with the unsupervised embedding as input, which means that properties which are relevant to the task

are likely better separated in the embedding. Similarly, as the neural network with a bag-of-words input (NNET-B) performed poorly, we can expect that properties that are separable in this representation will be less relevant to the task. As all of these representations performed similarly in the depth-3 decision trees, we can expect that they have each been able to identify the key properties that are particularly relevant to the class, e.g. one that corresponds to Horror, another to Comedy, and so on. So likely all embeddings share important properties relevant to the task, as well as introducing other properties.

The reason that NNET-U performed well in the neural network but poorly in the directions is likely because it did not disentangle properties such that those relevant to the class were grouped together. This means that although properties like "comedies" are more separable in the space, as they are not similar to the cluster-direction for "comedy", instead forming their own cluster-direction, the "comedies" cluster-feature is not used in the depth-3 decision tree, making it meaningless for the interpretable classifier but meaningful for the neural network. We find evidence to support this idea in Section 5.4.3.

In the case of movies the neural networks perform slightly better than the unsupervised representation when the unsupervised representation is used as input (NNET-U) but much worse when the PPMI bag-of-words is used as input (NNET-B). This is likely because the vocabulary for the movies is large (limited to 100,000 unique terms for movies from the original vocabulary size of 551,080 versus 51,064 for newsgroups, see Section 3.3) resulting in overfitting. It is interesting to see that for the place-types the single-term features and cluster-features for NNET-B outperformed the other embeddings, and for both the movies and the place-types the difference is minimal between the highest-scoring cluster-feature results on a depth-3 decision tree and the linear SVM on the unsupervised representation. This can lead us to assume that the properties obtained from the neural network embeddings are indeed meaningful.

NNET-B for the newsgroups outperformed the linear SVM with an unsupervised embedding by a large margin, and continued to do so for both the single-term features and the cluster-features. Interestingly, the cluster-features in a depth-3 decision tree also performed at the same level as the neural network. This can be understood to mean that the properties and subsequent cluster-features obtained from the NNET-B embedding must be particularly relevant to the task and also disentangled into concepts that directly correspond to the classes. Similarly the cluster-features for the place-types also performed at the same level as the neural network, but neither of them

performed better than the Linear SVM. This indicates that when learning to classify using a neural network starting from a bag-of-words, properties relevant to the task with good rankings of entities are identified in the embedding before the output layer.

In conclusion, disentangled properties that can be used to classify entities can be identified in the hidden layer before the output layer of neural networks, and in the case of newsgroups the interpretable features that correspond to these properties perform equivalent or better than the neural network. Some assumptions are made about what kind-of properties these embeddings represent, in particular that in neural networks that perform well embeddings will likely contain unique properties and also improved rankings for properties that are particularly relevant to the task. Additionally, for the NNET-B embedding in the newsgroups domain the properties will be disentangled.

### Direction results

As expected, the score of a depth-3 decision tree classifier when using the directions obtained from the neural network are close to the performance of the neural network.

There are a few reasons that a neural network representation would perform better than an unsupervised one, which are explained here with examples from the movies domain:

- Noise is no longer encoded spatially, making it easier to obtain meaningful features. (e.g. metadata like e-mail list names "listinfo" "robomod" are no longer highly separable, meaning that the only features found are meaningful)
- New features are encoded in the representation that contribute to the task (e.g. "Blood" may be in the unsupervised representation, corresponding to if the movie was funny, but in the neural network representation a feature for "Zombie" was introduced which helps classify if the movie is a horror)
- The properties that are relevant to the class are clustered together, resulting in a disentangled representation where features correspond more closely to the task, (e.g. The cluster-direction for "Blood, Gore" and the cluster-direction for "Scary, Terrifying" are instead clustered together to better represent the Horror genre in a single cluster-direction "Blood, Gore, Scary, Terrifying")



- The rankings for properties that are also encoded in the unsupervised representation are better (e.g. The "Comedy" direction more correctly ranks movies based on how funny they are)

If the directions perform well and use similar features it is reasonable to assume that the rankings are better than in the unsupervised representation. The next section qualitatively investigates directions in these embeddings.

### 5.4.3 Qualitative Investigation of Feedforward Networks

The qualitative section focuses on qualitatively investigating feed-forward neural network embeddings using the interpretable features. There are three main goals of this qualitative investigation section:

- Qualitatively investigate the interpretable feature representation obtained from the neural network embeddings.
- Qualitatively investigate the differences between between the embeddings.
- Qualitatively investigate the use of decision trees learned from the interpretable feature representation.

The first goal is simple to achieve: If the interpretable features obtained from the neural network makes intuitive sense, then it can be qualitatively investigating these features. The second goal is achieved by comparing the interpretable features obtained from the embeddings to the interpretable features obtained from the unsupervised space. Finally, decision trees are investigated by examining the properties that are used in them and their associated rankings.

When showing results for single directions, the words that these directions are labelled with e.g. "Blood" will be accompanied by the terms for two of its most similar directions e.g. "Blood (Gore, Horror)". This is to provide context for the word such that it can be understood more easily. Unlike the clustered features, the direction is not changed. Three embeddings are used in this section for each domain: The unsupervised embedding, the neural network embedding that

used the vector space as input, and the neural network embedding that used the bag-of-words as input.

Terms are scored using NDCG 4.5.2 as this has good results for a variety of tasks and spaces. The frequency threshold was set to the top 10,000 terms and the score threshold was set to the top 2,000 scoring terms in NDCG.

### **Top Scoring Terms**

In this section, we look at the top-scoring terms for three different domains. These are intended to give a general impression of what these terms and their associated similar directions are like, as well as get some initial insights into what the embeddings are representing.

In Table 5.2 the top NDCG scoring terms are listed for each domain and embedding type. One immediate observation is that the unsupervised embedding and the NNET with the unsupervised embedding as input ("NNET-U") have similar top scoring terms, e.g. "listinfo" "mailing". As expected, the directions including the noise in the original unsupervised embedding were not transformed significantly by the neural network. In the case of the neural network using BOW as input (NNET-B) however, these noisy terms are not present. This brings up the question, what exactly has the neural network learned such that it performs higher in F1-score in the case of the unsupervised embedding as input? In-order to further investigate this, terms that were unique to each representation were obtained. This was obtained using a basic procedure where if terms did not occur in either of the top 2,000 highest scoring terms for the other embeddings then they were considered not unique.

### **Common and Unique Terms to Each Embedding**

This section evaluates terms from the movies domain in terms of if they are unique to the embedding or common to all there embeddings. The intention is to see if the embeddings have introduced any new properties to the top 2,000.

A term is common if the term occurs in the 2,000 top scoring terms of both the other two embeddings. Interestingly, there were only 14 unique terms for the unsupervised embedding, and only 19 for the neural network that used the unsupervised embedding as input. Meanwhile,

## Movies

## Unsupervised Embedding

listinfo (mailman, rec)  
 robomod (mailman, rec)  
 mailing (mailman, listinfo)  
 noir (fatale, femme)  
 martial (kung, fight)  
 gay (homosexual, homosexuality)  
 horror (scary, scares)  
 prison (jail, prisoners)  
 arts (rec, listinfo)  
 musicals (musical, singing)  
 mailman (listinfo, robomod)

## Placetypes

southcoast (filters, reala)  
 interchange (underpass, hk)  
 canonrebel (1855mm, nikond300s)  
 statua (stern, 1st)  
 municipal (citizen, farbe)  
 madrid (noir, df)  
 reizen (seventies, canonef2470mmf28lsm)  
 commuter (underpass, muni)  
 crime (illegal, violence)  
 leafs (iris, cyan)

## Newsgroups

solid (design, single)  
 struggle (grew, landed)  
 spreading (pursued, tolerant)  
 salt (drinking, combinations)  
 random (attacks, described)  
 widely (developed, tend)  
 temperature (layers, consumption)  
 viable (motivation, emphasized)  
 vol (published, journal)  
 volt (garage, voltage)  
 voltage (amps, resistor)  
 volume (hundred, frequently)  
 volumes (historians, turkish)  
 voluntary (heterosexual, posed)  
 volvo (saab, chrysler)  
 tea (nagornokarabakh, mothers)  
 quick (careful, usual)  
 utterly (violates, sinners)

## NNET BOW Input

horror (pacing, dialog)  
 westerns (russian, digitally)  
 documentary (joke, ultimate)  
 comedies (actress, entertain)  
 hilarious (disappointment, dozen)  
 laughs (woods, rights)  
 sci (equivalent, intent)  
 adults (grown, rushed)  
 songs (battle, speak)  
 war (military, james)  
 western (don, stick)

leafs (botanic, f100)  
 aerialphotography (beaver, kiteaerialphotography)  
 irvine (jay, meet)  
 centralcoast (published, aloha)  
 swell (fin, polar)  
 pacificocean (puerto, waves)  
 trunks (birch, pair)  
 southflorida (fla, hawaiian)  
 lapland (alberi, topv333)  
 sunbathing (underwater, relaxed)

vol (vast, foot)  
 volt (amplifier, soldered)  
 voltage (cautious, scanned)  
 volume (fairly, distinguish)  
 volumes (expressed, scenario)  
 voluntary (demonstrating, explore)  
 volvo (aftermarket, horsepower)  
 tigers (carter, brady)  
 postage (straightforward, engaged)  
 povray (animations, pbmplus)  
 starters (worthwhile, fame)  
 secular (descendants, fundamentalists)  
 ring (increasing, behind)  
 utterly (des, retain)  
 occurring (communicate, contacted)  
 single (entire, prove)  
 oxygen (speeding, extraordinary)  
 widgets (shells, experimentation)

## NNET Embedding Input

listinfo (mailman, rec)  
 robomod (mailman, rec)  
 mailing (mailman, listinfo)  
 noir (fatale, femme)  
 horror (scary, horrific)  
 martial (kung, arts)  
 gay (homosexual, homosexuality)  
 prison (jail, convicted)  
 animation (animated, cartoon)  
 arts (rec, listinfo)  
 musicals (musical, numbers)  
 interchange (midday, elevated)  
 canonrebel (controluce, pigeons)  
 rave (dj, erin)  
 winnipeg (konicaminolta, twincities)  
 windmills (goldenhour, puffy)  
 statua (palacio, estatua)  
 reizen (1022mm, t3)  
 f456 (300mm, a550)  
 gibraltar (iow, upon)  
 song (singing, juni)

temperature (discoveries, surveys)  
 testify (ali, concentrated)  
 solid (company, sides)  
 tea (tech, buck)  
 widely (recently, 1982)  
 denial (judaism, kurds)  
 detecting (skeptical, signals)  
 dick (quoted, seen)  
 pitches (screw, headers)  
 random (dropped, atheism)  
 salt (stations, alike)  
 vol (contents, students)  
 volt (disagreement, clouds)  
 voltage (criteria, disclosed)  
 volume (although, quickly)  
 volumes (raster, josephus)  
 voluntary (transmitted, satan)  
 volvo (outlet, 302)

Table 5.2: NDCG Top-Scoring directions

## Movies

Unique Unsupervised	Unique NNET-B	Unique NNET-U	Common
immigrants (immigrant, america)	unlike (terrific, efforts)	carry (recent, hoping)	noir (fatale, femme)
flashback (flashbacks, present)	efforts (detail, directors)	federal (fbi, agents)	martial (kung, fight)
bloated (spectacle, overlong)	impression (throw, truth)	possessed (demonic, forces)	horror (scary, scares)
chapter (previous, installment)	viewed (catch, escape)	faustus (geocities, html)	arts (rec, robomod)
rebel (rebellious, freedom)	suggest (wondering, trip)	spike (african, lees)	musicals (musical, singing)
assault (attack, violent)	focuses (sadly, thoughts)	dashing (handsome, excitement)	hilarious (funniest, laughing)
client (lawyer, attorney)	terms (theatre, marvelous)	wartime (wwii, bombing)	westerns (western, cowboy)
predecessor (sequel, sequels)	suppose (heres, greatly)	phantom (sees, opera)	jokes (laughs, joke)
competitive (competition, sport)	credit (clever, sequence)	theories (theory, conspiracy)	romantic (romance, romances)
jealous (attraction, crush)	exception (beat, passed)	robots (sci, princess)	animation (animated, cartoon)
betrayal (loyalty, affair)	heres (suppose, negative)	abused (abuse, abusive)	western (westerns, west)
artsy (pretentious, artistic)	fare (twenty, concerned)	fiance (fianc, engaged)	songs (song, lyrics)
hotel (manager, vacation)	unable (convey, accept)	hatred (hate, racism)	comedies (comedic, laughs)
stereotypical (stereotypes, clich)	deliver (depth, limited)	mysteries (mystery, clues)	war (soldiers, military)

**Table 5.3: Terms from three different document embeddings, the unsupervised embedding, the neural network that used a bag-of-words as input and the neural network that used the unsupervised vector space as input. Arranged by NDCG, from highest to lowest..**

there were 770 unique terms for the neural network with bag-of-words as input. This shows that representing new directions was not the reason for superior performance for NNET-U. Rather, it is likely that terms that are relevant to the genre that were already in the top 2,000 top-scoring terms are given higher performance. In-order to qualitatively investigate further what has changed between the embeddings, the differences between the scores of the terms was obtained.

### The Difference Between The Term Scores In The Embeddings

In this section, the differences between the scores of terms in the movies domain are qualitatively analysed. The intention is to determine what some differences are between the unsupervised embeddings and the neural network embeddings, and evaluate previous assumptions regarding the behaviour of the network.

These results are for all term directions obtained from each embedding, so all 10,000 terms. In the first column of Table 5.4 the terms that the unsupervised representation has scored higher than NNET-U are not clearly related to the task of genres. For example, "yup", "wright", "wimpy" and "zoom" which are the top four highest score differences do not seem related to the

task at all. Meanwhile, the terms that the NNET-U embedding has gained in score are clearly relevant to the task, e.g. "animation", "adventure", "comedies". Although, it has also increased the separability of some noise e.g. the term "listinfo", which is a problem already identified when using the unsupervised representation as input to the neural network. This gives us some indication of what the neural network is doing to solve the task. This behaviour of the neural network decreasing separability of noisy terms and increasing separability of terms relevant to the task could contribute to the difference in score where the neural network outperformed a linear SVM on the unsupervised embedding.

Meanwhile NNET-B seems to have elevated terms that are not relevant to the task. As the neural network seems to have overfit due to the large vocabulary, it has made terms separable that are not immediately meaningful, e.g. "attends" "adds" "foray". However, as previously mentioned it did not have some of the same noise as the unsupervised embedding. However, it seems to have added its own noise.

Unsupervised - NNET-U	Unsupervised - NNET-B	NNET-U - Unsupervised	NNET-B - Unsupervised
zoom (cuts, editing)	zoom (cuts, editing)	allens (woody, allen)	atmosphere (mood, atmospheric)
wimpy (villian, instance)	zombies (zombie, undead)	animation (animated, cartoon)	biased (agree, showed)
wright (forrest, buyer)	williams (jim, includes)	adults (children, adult)	abomination (insult, horrible)
yards (shoot, yard)	willingly (explicitly, risk)	arts (rec, listinfo)	absent (absence, previous)
yup (junk, downright)	wrecked (crashes, crashed)	allen (woody, allens)	car (cars, driving)
watchers (underlying, holding)	welles (orson, kane)	adaptation (adaptations, adapted)	abandons (leaving, decides)
valiant (merit, admirable)	writers (write, finish)	animated (animation, cartoon)	attends (attending, attend)
wee (pee, pick)	willingness (conventional, goal)	aboard (ship, board)	adds (added, addition)
winding (road, wheel)	wars (enemy, war)	adventure (adventures, adventurous)	broaden (greater, balanced)
yea (hey, wanna)	winner (winning, won)	australian (australia, sydney)	artwork (painting, animation)
wax (inclined, usage)	walt (disneys, disney)	addiction (addict, addicted)	cheerfully (gleefully, bursts)
woke (yeah, wake)	unstable (psychotic, mentally)	berardinelli (employers, distributor)	foray (marks, venture)
wrecked (crashes, crashed)	walking (walk, walks)	abuse (abused, abusive)	aboard (ship, board)
winded (significant, deliver)	wrath (gods, angry)	actress (actresses, lady)	absurd (ridiculous, absurdity)
wine (bottle, drink)	widely (recently, equally)	apes (monkey, monkeys)	civilization (civilized, survival)
wrapping (product, needed)	wondering (figured, wanting)	comedies (comedic, laughs)	abrupt (lacked, ended)
yahoo (faustus, faust)	yawn (dull, tedious)	listinfo (mailman, rec)	active (helped, allowed)
whip (beat, kick)	zeal (enthusiasm, celluloid)	africa (african, countries)	admiration (respect, remarkable)
zeal (enthusiasm, celluloid)	wolf (wolves, pack)	bogart (casablanca, noir)	consciousness (profound, meaning)
zone (twilight, concept)	wire (wires, walls)	animals (animal, humans)	abandoned (deserted, forced)
wayward (cross, join)	undead (zombie, zombies)	anime (animation, japan)	attitude (respect, reasons)
visitor (arrival, visiting)	winter (snow, frozen)	african (racial, racist)	graphically (graphic, violent)
weaving (weaves, crafted)	yesteryear (throwback, nostalgic)	band (bands, rock)	acceptable (rhodes, internetreviews)
years (desires, affection)	voted (vote, awards)	accurate (accuracy, accurately)	advances (developed, attractive)
youngster (kid, childhood)	unquestionably (arguably, closing)	accuracy (accurate, inaccuracies)	depths (beneath, surface)
yell (angry, yelling)	wifes (husband, marriage)	artist (artists, artistic)	advice (decided, spent)
walker (damage, bill)	worthwhile (skip, include)	catholic (priest, church)	explanation (explained, explain)

**Table 5.4: The largest differences in term scores between embeddings for the movies. The first two columns are the terms that were most reduced in score, while the third and fourth columns are the terms that increased most in score.**

### **Newsgroups Decision Tree Cluster-Features**

In this section we qualitatively investigate the clusters for the newsgroups, with particular interest in why the NNET-B clusters performed as well as the neural network and better than the unsupervised representation by a wide margin.

In Table ??, we examine the clustered features used in the decision trees for the newsgroups. In particular, we obtain the top decision tree node for each class. This is to give an overview of the different kinds of clusters and how they perform for each task. When examining the difference between the cluster-features for the unsupervised embedding and NNET-B, we can see that these important features seem more relevant to the task. Rather than, refining the rankings of the previous properties, new cluster-directions have been found that more closely correspond to the task. For example, the cluster-feature used for comp.graphics contains many relevant terms "(pbmplus, sgis, sunview, phigs, pex, colormap, pixmap)" compared to the unsupervised embedding "(povray, raytracing)" as does the cluster-feature used for rec.autos "(sedan, camry, saab, ..., diesel, coupe)" versus "(sedan, camry)". Following this the conclusion is that the BOW representation simply better arranges the entities such that these natural cluster directions are meaningful to the class.

Strikingly, there is not a single clustered direction for NNET-U that is has combined more than a single term. This is because the cluster-directions for the newsgroups are such that there are many large cluster-directions full of noisy terms, and then many single-term cluster-features for the relevant terms. These larger cluster-features were not used in the depth-3 decision tree. However, this did not lead to good quantitative results. For example, "bikes" is particularly relevant to the class rec.motorcycles but the accompanying directions seen in the unsupervised and NNET-B embeddings are not present. This is likely due to the hyper-parameter for the clusters, but it is interesting as that means these single-directions performed better than clustering terms together.

Newsgroups	Unsupervised Embedding	NNET-B	NNET-U
alt.atheism	(celestial, creationist, psalm, ..., fallacy, baptism)	(messenger, faiths)	(homosexuality)
comp.graphics	(povray, raytracing)	(pbmplus, sgis, sunview, phigs, pex, colormap, pixmap)	(tiff)
comp.os.ms-windows.misc	(winini, systemini, cica)	(cirrus, workgroups, hicolor, ..., keystrokes, sdk)	(xlib)
comp.sys.ibm.pc.hardware	(lpt1, irq, chipset, mfm, logitech)	(qemm, emm386, hernandez, ..., com2, brewers)	(motherboards)
comp.sys.mac.hardware	(centris, lciii)	(democrats, emissions, helicopters, enforced, tribes, reign, clintons)	(quadra)
comp.windows.x	(xdm, makefile, r4, ..., rainer, ow)	(xmotif, tvtwm, xfree86, ..., xsun, polygon)	(xlib)
misc.forsale	(diffiehellman, exterior, strips, ..., uuep, slots)	(267, saga, warriors, ..., pov, carnage)	(obo)
rec.autos	(sedan, camry)	(sedan, camry, saab, ..., diesel, coupe)	(toyota)
rec.motorcycles	(porsche, countersteering, msf)	(carb, bikes, reed)	(bikes)
rec.sport.baseball	(gant, duke, padres, ..., marlins, slg)	(larkin, marlins, pennant, ..., platoon, coaching)	(pitching)
rec.sport.hockey	(providence, keenan, hawks, ..., champs, ahl)	(nyr, messier, motto, ..., goaltending, keenan)	(stanley)
sci.crypt	(playback, ciphertext, cryptanalysis, ..., escrow, nist)	(authentication, diffiehellman, publickey, 80bit, vesselin, skipjack)	(rsa)
sci.electronics	(ham, amp, reactor, watts, amplifier, amps)	(omega, diode, charger, ..., antennas, joystick)	(voltage)
sci.med	(intake, calcium, antibiotic, ..., kidney, quack)	(bethesda, sensation, defects, ..., therapies, tablets)	(patients)
sci.space	(reboost, fusion, astronomers, ..., pasadena, galaxy)	(shafer, reusable, pluto, ..., orbiter, billboards)	(lunar)
soc.religion.christian	(divorce, sinless, mooses, ..., corinthians, baptized)	(pastor, baptized, congregation, corinthians, repent, pagan)	(scripture)
talk.politics.guns	(federalist, unregistered, tyranny, ..., wright, shotguns)	(tpg, progun, 9mm, ..., rkba, shotguns)	(batf)
talk.politics.mideast	(arab, rebellion, syrian, ..., zionist, gaza)	(asalasdpaarf, revisionism, zionism, ..., hamas, grandparents)	(palestinian)
talk.politics.misc	(croats, redraw, shelling, ..., bosnians, yugoslavia)	(wiretapping, safeguards, visi)	(homosexual)
talk.religion.misc	(divorce, sinless, mooses, ..., corinthians, baptized)	(thy, forgiveness, churchs, ..., protestant, ephesians)	(scripture)

Table 5.5: The top node of the Decision Tree for newsgroups



### **Movies Decision Tree Cluster-Features**

In this section, we qualitatively investigate the cluster-features used in the top decision tree nodes for the movies domain, with a particular interest in why all embeddings have similar overall predictive performance despite containing very different properties (as talked about in Section 5.4.3).

One observation that follows the discussion in Section 5.4.3 is that the Adventure genre is classified using a cluster-feature that contains the word "adventure" in the case of NNET-U (adventure, spectacle, exciting, ..., seat, boyfriend), rather than the "(animation, animated, anime, voiced, cartoon)" cluster-feature that is re-used in the unsupervised embedding. This follows the previous observation as the term "adventure" was one of the biggest differences between the unsupervised representation and NNET-U, where it had increased in NDCG score for NNET-U. However, despite this the performance overall is the same.

The reason for the similar performance is likely that at the cost of re-organizing the representation such that the adventure cluster-direction existed, other cluster-directions were disrupted. For example the history class is classified in the unsupervised representation by the seemingly relevant "(events, accuracy, accurate, facts, confusing)", but in the case of NNET-U this cluster is disrupted by many seemingly noisy terms "(western, historical, musicians, ..., biography, propaganda)". Note that we do not really see this in the newsgroups Table for NNET-B, where instead new and meaningful cluster-features are found that are relevant to the task.

Movies	Unsupervised Embedding	NNET Bow	NNET Embedding
Action	(fight, fighting, epic, ..., battle, weapons)	(martial)	(martial, fight, fighting, fights, choreography, choreographed, fighter)
Adventure	(animation, animated, anime, voiced, cartoon)	(adventure, adventures)	(adventure, spectacle, exciting, ..., seat, boyfriend)
Animation	(animation, animated, anime, voiced, cartoon)	(voiced)	(voice, voiced, recording, voices, vocal, listening)
Biography	(biography, biopic)	(biopic)	(gritty, historically, fiction, ..., accurate, accuracy)
Comedy	(hilarious, jokes, comedies, ..., funniest, funnier)	(comedies, funniest, funnier, ..., laughed, slapstick)	(hilarious, jokes, comedies, ..., funniest, funnier)
Crime	(noir, crime, caper, criminal, criminals, crimes)	(crime, police, criminal, cop, cops)	(gangster, gangsters)
Documentary	(documentary, footage, documentaries, interviews, interviewed, informative)	(documentary, footage, documentaries)	(documentary, footage, documentaries, ..., extras, interviewed)
Drama	(emotional, magical, waves, ..., silent, evidence)	(emotional, emotions, emotionally, ..., relationships, study)	(bond, emotional, families, ..., powerful, gripping)
Family	(adults, children, childreans, ..., parents, daughter)	(adult, ages, children, parents, teenager)	(disneys, walt)
Fantasy	(fantasy, fairy, fairytale)	(magic, colorful, comical, lavish, delightfully, dazzling, colors)	(fantasy, surreal, elm, fairy, fairytale, dreams, dream)
Film-Noir	(femme, fatale)	(noir, vintage)	(femme, fatale)
History	(events, accuracy, accurate, facts, confusing)	(historically)	(western, historical, musicians, ..., biography, propaganda)
Horror	(horror, creepy, slasher, ..., scares, scare)	(creepy, scare, spooky, scary, monster, menacing, nightmares)	(horror, creepy, scares, ..., spooky, chills)
Music	(musicals, songs, singing, ..., song, numbers)	(songs, musical, song, ..., sing, dancing)	(songs, singing, song, ..., sings, singer)
Musical	(musicals, songs, singing, ..., song, numbers)	(musicals, Broadway)	(songs, singing, song, ..., sings, singer)
Mystery	(detective, investigation, mystery, clues, mysterious, investigating)	(mystery, dramatic, fiction, ..., psychological, slowly)	(thriller, suspense, hitchcock, ..., thrillers, suspenseful)
Romance	(romantic, charming, romance, ..., charm, chick)	(romance, romantic, chemistry, ..., handsome, scenery)	(wedding, marry, marries, marrying, bride)
Sci-Fi	(sci, alien, space, aliens, outer, invasion)	(sci, science, futuristic)	(science, scientific, scientist, ..., scientists, investigating)
Short	(episodes, episode, seasons, aired, television, storylines) (python, monty)	(agree, soundtrack, product, ..., happening, inside)	(geocities, html, aol, faustus)
Sport	(coach, sports, team, sport, football)	(sports, sport, coach)	(boxing)
Thriller	(thriller, suspense, adventure, ..., suspenseful, tension)	(thriller, suspense, thrillers, suspenseful, thrills, thrilling)	(thriller, suspense, hitchcock, ..., thrillers, suspenseful)
War	(war, soldiers, vietnam, ..., military, troops)	(wwii, german)	(war, soldiers, vietnam, ..., military, troops)
Western	(westerns, western)	(westerns)	(outlaw)

Table 5.6: The top node of the Decision Tree for Movies

### Top-Ranking Entities On Decision Tree Cluster-Features in the Place-types Domain

The intention of this section is to investigate the rankings of entities in the embeddings and see if they make sense. The intuition is that despite neural network embeddings obtaining directions that make reasonable sense, their entities may be disrupted. This is determined by taking the top 5 entities for each top node cluster-feature from decision trees as in the previous section. Note that it may not seem meaningful to take only the top 5 entities, but as there are only 391 entities overall for the class, and very few of those are positive instances of the class, they are actually extremely relevant. Essentially, if the entities do not make sense for the cluster directions in the neural network embedding but do for the unsupervised embedding, it is reasonable to say that the interpretability of the features is not as good as in an unsupervised representation.

To begin, we look at the clusters used to classify "CollegeAndUniversity", the unsupervised representation uses the cluster "(annarbor, graduation, eugene, institute, highschool, cal)" that seems particularly relevant to educational institutions, and its associated entities also seem very meaningful and relevant "(college, campus, college campus, university, school)". For NNET-B, the cluster itself seems absurd "(investment)", but the associated entities are relevant to that cluster "(commercial real estate, large construction, retirement home, ..., rental property)". Similarly, the cluster for NNET-U does not seem relevant to the class and neither do its top entities "(snowboard, turism, amusementpark, ..., waltdisney, disneyworld)" "(video game store, theme park, space shuttle, launch pad, speedway)". So in this case, the entities seem to hold.

In general, the results for this table follow a similar trend. Entities align with the meaning of cluster-features, even in neural network embeddings. The problems of classification come from poor selection of these cluster-features, rather than the cluster-features of those entities being incorrect. There is an interesting difference in the case of classifying NightLifeSpot, where three distinct but relevant clusters and associated entities are found and used in each of the embeddings. In the unsupervised representation, a cluster-feature related to music was used "(audience, instrument, musicians, ..., dancers, bands)" with relevant entities for that "(stage, bar, rock, sound, music venue)", meanwhile in the case of NNET-B, a cluster for drinking alcohol was used instead "(booze, vodka, whiskey, liquor)" with top entities of places to get that alcohol "(dive bar, beer garden, karaoke bar, hotel bar, cocktail bar)", finally the NNET-U identifies a cluster more related to sex but also including tattoo and dance studios (shoulder, darkroom,

topless, ..., boobs, lips) with associated entities "(dance studio, tattoo studio, shoulder, topless beach, strip club)".

In conclusion for this section, entities seem to make sense in the case of place-types where they align well with their cluster-feature. Additionally, combining the use of a Decision Tree, the associated cluster terms with the cluster-feature, as well as the top entities gives great insight into the representation and what it is doing, as each provides valuable context for each other.

Class	Decision Tree Cluster-Feature	Top 5 Entities
ArtsAndEntertainment	Unsupervised	(restaurant, market, asian restaurant, seafood restaurant, japanese restaurant)
CollegeAndUniversity	(snack, vegetable, lemon, pepper, vegetables)	(college, campus, college campus, university, school)
Food	(annarbor, graduation, eugene, institute, highschool, cal)	(restaurant, asian restaurant, japanese restaurant, seafood restaurant, chinese restaurant)
NightlifeSpot	(cuisine)	(stage, bar, rock, sound, music venue)
ParksAndOutdoors	(audience, instrument, musicians, ..., dancers, bands)	(coast, shore, coastline, shoreline, beach)
ProfessionalAndOtherPlaces	(cloudscape, solitary, hazy, ..., fluffy, intense)	(court house, courthouse, government building, state, cemetery)
Residence	(nationalhistoriclandmark, register, revival, jefferson, independence, pioneer, civilwar)	(house, home, residential building, apartment, historic building)
ShopsAndService	(laundry, dwelling)	(market, boutique, store, clothing store, vintage store)
TravelAndTransport	(mannequin, etsy, crafts, boutique, bags, jewelry, necklace)	(market, desert, national park, crater, volcano)
ArtsAndEntertainment	(donkey, costarica, tanzania, ..., reisen, andes)	(sugar mill, silo, corral, abandoned farm, ranch)
CollegeAndUniversity	NNET-B	(commercial real estate, large construction, retirement home, ..., rental property)
Food	(ruraldecay)	(football field, training camp, strait, baseball stadium, continent)
NightlifeSpot	(investment)	(dive bar, beer garden, karaoke bar, hotel bar, cocktail bar)
ParksAndOutdoors	(dolphins)	(piano bar, public housing, waiting room, residential street, dining room)
ProfessionalAndOtherPlaces	(booze, vodka, whiskey, liquor)	(sand bar, vineyard, monsoon forest, ski lodge, topless beach)
Residence	(condos, venice, entertainment, ..., socks, americanflag)	(battlefield, national monument, battlefield park, cliff dwelling, space shuttle)
ShopsAndService	(gardening, tiki, cocktail, ..., candle, horticulture)	(diocese, convent, tomb, college theater, study)
TravelAndTransport	(tribute, plains, des, acropolis, aa)	(railroad tunnel, terminal, airport tram, railroad signal, railroad yard)
ArtsAndEntertainment	(cloister)	(cubicle, newsroom, workroom, bedroom closet, detached house)
CollegeAndUniversity	(dmctz3, panasonicdmctz3, panasonicitz3)	(video game store, theme park, space shuttle, launch pad, speedway)
Food	NNET-U	(molecular gastronomy restaurant, whisky bar, gourmet shop, cocktail bar, juice bar)
NightlifeSpot	(messy, schoolhouse, beatles, ..., halftimbered, publictransport)	(dance studio, tattoo studio, shoulder, topless beach, strip club)
ParksAndOutdoors	(snowboard, tourism, amusementpark, ..., waltdisney, disneyworld)	(rookery, hunting reserve, wetland, wildlife reserve, nest)
ProfessionalAndOtherPlaces	(boutique, liquor, olive, ..., counter, dessert)	(retail outlet, electronics store, tourist information center, jewelry store, souk)
Residence	(shoulder, darkroom, topless, ..., boobs, lips)	(ventilation shaft, abandoned prison, sanatorium, abandoned complex, reformatory)
ShopsAndService	(nesting, tits, mouth, ..., gulls, avian)	(video game store, jewelry store, thrift store, workroom, nail salon)
TravelAndTransport	(islam, muslim, supermarket, ..., sales, marruecos)	(crop farm, flowerbed, tongue, nest, crop circle)
	(heath, ward, lock, ..., stained, argyll)	
	(video, candy, clothing, ..., skin, blonde)	
	(survey, mammal, antenna, ..., paws, fur)	

Table 5.7: The placetypes clusters used at the top of the decision tree and the associated top-ranked entities

### Top Entities for Common Terms

Despite the previous section giving insight into the cluster-features by providing context with the top 5 entities, it did not illustrate the differences in entity rankings between the embeddings for the same properties. Instead, it showed that different embeddings used different properties as the top node of the decision tree for a class, often covering different facets in different embeddings. In this section, we investigate the differences between top entities for common terms. The method is the same as Section 5.4.3 to obtain these common terms, but instead of taking the top 2,000 terms we take the top 500, so as not to include low-scoring terms that will have poor entity representations across the embeddings.

Interestingly, the resulting terms have very common top entities in their rankings, including the neural network representation that used the bag-of-words as input. This further validates the idea given earlier that the representations are able to achieve common scores when using direction and cluster features because they find common terms that can be used in the classes. Essentially, they are able to find the same relevant terms that matter for classification.

## 5.5 Qualitative Evaluation of Auto-encoders

We base our experiments on the movie reviews domain. To collect the terms that are likely to correspond to property names, we collect adjectives and nouns that occur at least 200 times in the movie review data set, collecting 17,840 terms overall. These parameters are used for all entity embeddings induced by the auto-encoder.

### 5.5.1 Software, Architecture and Settings

To implement the denoising auto-encoders, the Keras [?] library is used. As in 6, scikit-learn [?] is used for the SVM implementation. All of the code and data for this Chapter is available on GitHub<sup>1</sup>. A 200 dimensional MDS space is used as the input to our stack of auto-encoders, as this performed the best in Chapter 6. The network is trained using stochastic gradient descent and the mean squared error loss function. For the encoders and decoders, the tanh activation

---

<sup>1</sup><https://github.com/eygrr/RulesFromAuto-encoders>

Cluster Features	Top 5 Ranked Entities
Unsupervised Embedding	
bass (drums, tom)	(indie theater, jazz club, music studio, music venue, rock club)
arrival (ticket, departure)	(taxiway, airport tram, airport, aircraft cabin, airport gate)
condo (condominium, apartments)	(rental property, condominium, condo, villa, plaza)
palazzo (ff, notte)	(campanile, palace, villa, triumphal arch, cuesta)
fans (ball, player)	(basketball stadium, hockey arena, soccer stadium, baseball stadium, football stadium)
animalplanet (feeder, natureselegantshots)	(zoo, zoological garden, nest, moais, animal shelter)
rent (homes, villas)	(rental property, real estate offices, cuesta, condo, condominium)
agriculture (agricultural, fields)	(cropland, conifer forest, olive grove, arboretum, sugar plantation)
champions (win, league)	(football stadium, soccer stadium, basketball stadium, college stadium, cuesta)
cielo (azul, nubes)	(cuesta, campanile, plaza, villa, playa)
drummer (drums, who)	(jazz club, cuesta, moais, music venue, indie theater)
accommodation (accomodation, guest)	(rental property, amenities, hotel pool, suite, ski chalet)
decayed (forgotten, exploring)	(abandoned airfield, sanatorium, concentration camp, abandoned prison, hospital ward)
yankees (louis, victory)	(baseball stadium, basketball stadium, baseball field, hockey arena, stadium)
NNET-B	
bass (drums, tom)	(indie theater, jazz club, music studio, music venue, rock club)
arrival (ticket, departure)	(taxiway, airport tram, airport, aircraft cabin, airport gate)
condo (condominium, apartments)	(rental property, condominium, condo, villa, plaza)
palazzo (ff, notte)	(campanile, palace, villa, triumphal arch, cuesta)
fans (ball, player)	(basketball stadium, hockey arena, soccer stadium, baseball stadium, football stadium)
animalplanet (feeder, natureselegantshots)	(zoo, zoological garden, nest, moais, animal shelter)
rent (homes, villas)	(rental property, real estate offices, cuesta, condo, condominium)
agriculture (agricultural, fields)	(cropland, conifer forest, olive grove, arboretum, sugar plantation)
champions (win, league)	(football stadium, soccer stadium, basketball stadium, college stadium, cuesta)
cielo (azul, nubes)	(cuesta, campanile, plaza, villa, playa)
drummer (drums, who)	(jazz club, cuesta, moais, music venue, indie theater)
accommodation (accomodation, guest)	(rental property, amenities, hotel pool, suite, ski chalet)
decayed (forgotten, exploring)	(abandoned airfield, sanatorium, concentration camp, abandoned prison, hospital ward)
yankees (louis, victory)	(baseball stadium, basketball stadium, baseball field, hockey arena, stadium)
NNET-U	
bass (tom, drums)	(music studio, jazz club, rock club, music school, piano bar)
arrival (departure, journey)	(airport tram, airport control tower, airport, airport lounge, toll booth)
condo (condominium, condominiums)	(condo, condominium, rental property, suite, hotel pool)
palazzo (palais, efs)	(campanile, palace, cathedral, villa, plaza)
fans (arena, name)	(football stadium, basketball stadium, soccer stadium, hockey arena, stadium)
animalplanet (tiere, vosplusbellesphotos)	(zoo, zoological garden, nest, animal shelter, rookery)
rent (rental, animalplanet)	(rental property, condo, suite, condominium, real estate offices)
agriculture (farming, petrol)	(crop farm, olive grove, arboretum, cropland, wheatfield)
champions (win, victory)	(football stadium, soccer stadium, stadium, basketball stadium, hockey arena)
cielo (ciel, nubes)	(campanile, cuesta, cathedral, plaza, villa)
drummer (guitarist, drums)	(jazz club, rock club, stage, piano bar, music venue)
accommodation (condominium, hostel)	(amenities, rental property, suite, hotel pool, resort)
decayed (forgotten, decaying)	(sanatorium, abandoned airfield, barracks, abbey, military barracks)
yankees (mlb, league)	(baseball stadium, baseball field, avenue, basketball stadium, hockey arena)

**Table 5.8: Terms common for all embeddings and the associated ranking of entities on those term directions for each embedding. Arranged by NDCG-score in the unsupervised embedding..**

function is used. For the first auto-encoder, the same size layer as the input is maintained. Afterwards, the hidden representation size is halved each time it is used as input to another auto-encoder, and this process is repeated three times, resulting in four new hidden representations  $\{Input : 200, Hidden : 200, 100, 50, 25\}$ . The input space is corrupted using Gaussian noise with a standard deviation of 0.6 each time. As the lower layers are closer to the bag-of-words representation and are higher dimensional, the Kappa scores are higher in earlier spaces, as it is easier to separate entities. We address this in the clusters by setting the high Kappa score threshold  $T^+$  such that the number of terms we choose from is twice the number of dimensions in the space. Similarly, we set  $T^-$  such that 12,000 directions are available to assign to the cluster centres in every space.

### 5.5.2 Qualitative Evaluation of Induced Clusters

In Table ??, we illustrate the differences between clusters obtained using standard auto-encoders and denoising auto-encoders. Layer 1 refers to the hidden representation of the first auto-encoder, and Layer 4 refers to the hidden representation of the final auto-encoder. As single labels can lead to ambiguity, in Table 1 we label clusters using the top three highest scoring terms in the cluster. Clusters are arranged from highest to lowest Kappa score.

Both auto-encoders model increasingly general properties, but the properties obtained when using denoising auto-encoder properties are more general. For example, the normal auto-encoder contains properties like "Horror" and "Thriller", but does not contain more general properties like "Society" and "Relationship". Further, "Gore" has the most similar properties "Zombie" and "Zombies" in Layer 1, and has the most similar properties of "Budget" and "Effects" in Layer 4. By representing a category of movie where "Budget" and "Effects" are important, the property is more general.

### 5.5.3 Qualitative Evaluation of Induced Symbolic Rules

Our aim in this work is to derive symbolic rules that can be used to explain the semantic relationships between properties derived from increasingly general entity embeddings. We provide examples of such rules in this section. Since the number of all induced rules is large, here we



Table 5.9: A comparison between the first layers and the fourth layers of two different kinds of auto-encoders.

Standard Auto-encoder		Denoising Auto-encoder	
Layer 1	Layer 4	Layer 1	Layer 4
horror: terror, horrific	horror: victims, nudity	gore: zombie, zombies	society: view, understand
thriller: thrillers, noir	documentary: perspective, insight	jokes: chuckle, fart	emotional: insight, portrays
comedies: comedy, timing	blood: killing, effects	horror: terror, horrific	stupid: flick, silly
adults: disney, childrens	suspense: mysterious, tense	emotionally: tragic, strength	gore: budget, effects
husband: wife, husbands	thriller: thrillers, cop	gags: zany, parodies	military: war, ship
relationships: intimate, angst	gory: gruesome, zombie	hindi: bollywood, indian	romance: younger, handsome
nudity: naked, gratuitous	beautifully: satisfying, brilliantly	touching: teach, relate	ridiculous: awful, worse
political: politics, nation	emotional: complex, struggle	scary: frightening, terrifying	government: technology, footage
smart: slick, sophisticated	laughed: laughing, loud	documentary: document, narration	awesome: chick, looked
creepy: sinister, atmospheric	charming: delightful, loves	adults: disney, teaches	political: country, documentary
laughed: humorous, offensive	hilarious: funny, parody	laughed: brow, laughter	relationship: relationships, sensitive
adventure: adventures, ship	scares: halloween, slasher	thriller: thrillers, procedural	horror: genre, dark
actions: reaction, innocent	funniest: funnier, gags	cgi: animated, animation	waste: concept, plain
cute: adorable, rom	emotions: respect, relationships	suspense: clues, atmospheric	army: disc, studio
british: england, accent	laugh: mom, crazy	dumb: mindless, car	combat: enemy, weapons
horrible: worse, cheap	filmmaker: approach, artist	political: propaganda, citizens	supporting: office, married
narrative: filmmaker, structure	drama: portrayed, portrayal	witty: delightfully, sarcastic	amazon: bought, copy
digital: dolby, definition	interviews: included, showed	laughing: outrageous, mouthed	study: details, detail
gory: graphic, gruesome	comedic: comedies, humorous	relationships: ensemble, interactions	land: water, super
romantic: handsome, attractive	emotionally: central, relationships	creepy: mysterious, eerie	chemistry, comedies, comedic

only show high accuracy rules that cover 200 samples or more. Still, we naturally cannot list even all the accurate rules covering more than 200 samples. Therefore we focus here on the rules which are either interesting in their own right or exhibit interesting properties, strengths or limitations of the proposed approach. The complete list of induced rules is available online from our GitHub repository<sup>2</sup>.

For easier readability, we post-process the induced rules. For instance, the following is a rule obtained for the property “Gore” in the third layer of the network shown in the original format produced by JRip:

```
IF scares-L2 <= 6 AND blood-L2 <= 8 AND funniest-L2 >= 22
=> classification=+ (391.0/61.0)
```

In this rule, `scares-L2 <= 6` denotes the condition that the movie is in the top 6% of rankings for the property “scares” derived from the hidden representation of the second auto-encoder. We will write such conditions simply as “`Scares2`”. Similarly, a condition such as `funniest-L2 >= 22`, which indicates that the property is not in the top 22%, will be written as `âNOT Funniest2â`. In this simpler notation the above rule will look as follows:

```
IF Scares2 AND Blood2 AND NOT Funniest2 THEN Gore3
```

This rule demonstrates an interpretable relationship. However, we have observed that the meaning of a rule may not be clear from the property labels that are automatically selected. In such cases, it is beneficial to label them by including the most similar cluster terms. For example, using the cluster terms below we can see that “Flick” relates to “chick-flicks” and that “Amazon” relates to old movies:

```
IF Flick2 AND Sexual2 AND Cheesy2 AND NOT Amazon2 THEN Nudity3
```

```
Flick2: {Flicks, Chick, Hot}
```

```
Amazon2: {Vhs, Copy, Ago}
```

Rules derived from later layers use properties described by rules from previous layers. By seeing rules from earlier layers that contain properties in later layers, we can better understand

<sup>2</sup><https://github.com/eygrr/RulesFromAuto-encoders>

what the components of later rules mean. Below, we have provided rules to explain the origins of components in a later rule:

```
IF Emotions2 AND Actions2 THEN Emotions3
IF Emotions2 AND Emotion2 AND Impact2 THEN Journey3
IF Emotions3 AND Journey3 THEN Adventure4
```

We observe a general trend that as the size of the representations decreases and the entity embeddings become smaller, rules have fewer conditions, resulting in overall higher scoring and more interpretable rules. To illustrate this, we compare rules from an earlier layer to similar rules in a later layer:

```
IF Romance1 AND Poignant1 AND NOT English1 AND NOT French1
AND NOT Gags1 AND NOT Disc1 THEN Relationships2

IF Relationships2 AND Emotions2 AND Chemistry2 THEN Romantic3
IF Emotions2 AND Compelling2 THEN Beautifully3
IF Warm2 AND Emotions2 THEN Charming3
IF Emotions2 AND Compelling2 THEN Emotional3
```

Rules in later layers also made effective use of a NOT component. Below, we demonstrate some of those rules:

```
IF Touching3 AND Emotions3 AND NOT Unfunny3 THEN Relationship4
IF Laughs3 AND Laugh3 AND NOT Compelling3 THEN Stupid4
IF Touching3 AND Social3 AND NOT Slasher3 THEN Touching4
```

As the same terms were used to find new properties for each space, the obtained rules sometimes use duplicate property names in their components. As the properties from later layers are a combination of properties from earlier layers, the properties in later layers are refinements of the earlier properties, despite having the same term. Below, we provide some examples to illustrate this:

```
IF Emotions2 AND Actions2 THEN Emotions3
```

```
Emotions2: {Acted, Feelings, Mature}
```

Actions<sub>2</sub>: {Control, Crime, Force}

Emotions<sub>3</sub>: {Emotion, Issue, Choices}

IF Horror<sub>2</sub> AND Creepy<sub>2</sub> AND Scares<sub>2</sub> THEN Horror<sub>3</sub>

Horror<sub>2</sub>: {Terror, Horrific, Exploitation}

Creepy<sub>2</sub>: {Mysterious, Twisted, Psycho}

Scares<sub>2</sub>: {Slasher, Supernatural, Halloween}

Horror<sub>3</sub>: {Creepy, Dark, Chilling}

IF Touching<sub>2</sub> AND Chemistry<sub>2</sub> THEN Touching<sub>3</sub>

IF Touching<sub>2</sub> AND Emotions<sub>2</sub> THEN Touching<sub>3</sub>

IF Compelling<sub>2</sub> AND Emotional<sub>2</sub> AND Suspense<sub>2</sub> THEN Compelling<sub>3</sub>

IF Romance<sub>2</sub> AND Touching<sub>2</sub> AND Chemistry<sub>2</sub> THEN Romance<sub>3</sub>

IF Emotionally<sub>2</sub> AND Emotions<sub>2</sub> AND Compelling<sub>2</sub> THEN Emotionally<sub>3</sub>

## 5.6 Conclusion

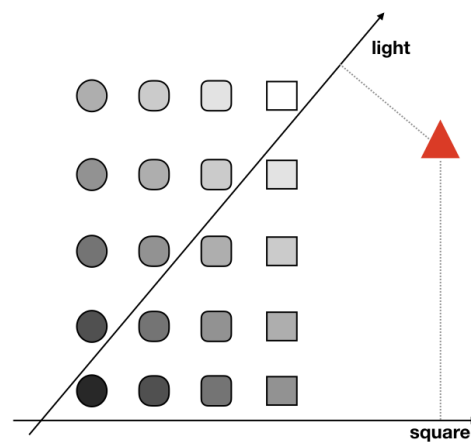
Feedforward neural networks and auto-encoders are qualitatively investigated using the methods outlined in Chapter 6. These properties are meaningful and interpretable in both representations, and in the case of the auto-encoders as the space size reduces the representation becomes more abstract. Through qualitative investigation rules show some promise in being used to link together properties in the layers and form some explanation of abstraction. Depth-3 decision trees on the final hidden layer of the neural network are found to retain some or all of the performance of the original neural network and also remain interpretable. For future work, the interpretability of these disentangled representations could be compared to alternative approaches, and they could be used to benefit those that work on those safety and fairness of neural network models, in particular to make black-box state-of-the-art models transparent by explaining layer-to-layer connections.

# Fine-tuning Vector Spaces to Improve Their Directions

## 6.1 Introduction

Chapter ?? introduced a method to obtain feature-directions from off-the-shelf vector-spaces, as well as methods to test the quality of these feature-directions and their associated feature-rankings. Then, this method was applied in Chapter 5 to obtain feature-directions from the layers of neural networks. However, feature-directions obtained from either of these vector spaces can sometimes be sub-optimal. For example in the case of neural network auto-encoders, it was found that the quality of feature-directions in a auto-encoder representation degrade from a maximal Kappa score of 0.52 in the initial layer of to a maximal Kappa score of 0.18 on the 5th layer (See Section ??).

In figure 6.1, a problem that can occur with feature-directions from representations learned with a similarity-centred objective, e.g. Multi-Dimensional Scaling (see Section 2.4.9) is illustrated. This is an example problem in the toy domain of shapes, where basic geometric shapes are embedded in a two-dimensional space. In this example, directions have been identified which encode how light an object is and how closely its shape resembles a square. While most of the shapes embedded in this space are grey-scale circles and squares, one of the shapes embedded in this space is a red triangle, a clear outlier. When considering that the objective the space is learned with is similarity, the spatial representation for this triangle is correct, as it is far from all the other shapes. However, when ranking the shapes on the feature-directions for square and light, the outlier takes up an extreme position on the rankings. This means that the triangle is ranked incorrectly, as it is considered to be the shape that most exhibits the features “light” and



**Figure 6.1:** Toy example showing the effect of outliers in a two-dimensional embedding of geometric shapes..

“square”.

Ideally, representations would be learned with knowledge of feature-directions. For example, the method to learn a representation of the toy domain would know that it should model the features “square” and “light” rather than a similarity objective, so that this triangle would end up closer to the bottom-left corner. However, as we cannot a-priori determine the features the space must be learned from, it is difficult to learn a representation in this way. This Chapter instead introduces an unsupervised method that given a representation and its associated feature-directions, we can obtain a vector space and associated feature-directions where the quality of the feature-directions is prioritized over the existing structure. The intention of this method is to resolve issues like those described in the previous two paragraphs.

To introduce the idea behind the method, we start with the assumption that each feature-direction has one feature-word, which describes the feature: if the feature-ranking of a document on a feature-direction is faithful to the bag-of-words score for the feature-word in the document, then the feature-ranking is good. To give an example of why this assumption is useful, in the IMDB movies domain 3.2 Multi-Dimensional Scaling (MDS) spaces there is the case of an Indian Bollywood movie that is very unlike other movies, as its reviews only use language specific to Bollywood films and the amount of reviews it has is low overall. This movie occupies a top-ranking position in a variety of feature-directions, as a consequence of it being very dissimilar to other movies. The fine-tuning process solves this problem by attempting to match its ranking in the vector space that is very high, to its bag-of-words value, which is zero. This results in this

Feature direction	Highest ranking objects	Highest fine-tuned ranking objects
{steep, climb, slope}	mountain, landscape, national park	ski slope, steep slope, slope
{illuminated, illumination, skyscraper}	building, city, skyscraper	tall building, office building, large building
{play, kid, kids}	school, field, fence	college classroom, classroom, school
{spooky, creepy, scary}	hallway, fence, building	hospital room, hospital ward, patient room
{amazing, dream, awesome}	fence, building, beach	hotel pool, resort, beach resort
{pavement, streetlight, streets}	sidewalk, fence, building	overpass road, overpass, road junction
{dead, hole, death}	fence, steps, park	grave, cemetery, graveyard
{spire, belltower, towers}	building, arch, house	bell tower, arch, religious site
{stones, moss, worldheritage}	landscape, fence, steps	ancient site, ancient wall, tomb
{mosaic, tile, bronze}	building, city, steps	cathedral, church, religious site

**Table 6.1: Comparing the highest ranking place-type objects in the original and fine-tuned space. .**

obscure outlier movie being moved down drastically in the rankings.

To give some real examples, in Table 6.1, names of documents are shown ranked on feature directions in the domain of place-types (See Section 3.2). In these examples, for the cluster-feature *{steep, climb, slope}*, the top ranked document *mountain* is clearly relevant. However, the next two documents — *landscape* and *national park* — are not directly related to this feature. Intuitively, they are ranked highly because of their similarity to *mountain* in the vector space. Similarly, for the second feature, *building* is ranked highly because of its similarity to *skyscraper*, despite intuitively not having this feature. Finally, *fence* received a high rank for several features, mostly because it is an outlier in the space.

Generally, the method that fine-tunes vector spaces and their associated feature-directions is as follows: First, a vector space is learned from bag-of-words representations of the considered documents, using a standard similarity-centric method or neural network. Next, the method from Chapter ?? is used to obtain feature-directions and their associated words from a vector space. Then, following our assumption outlined in the previous paragraph, documents are ranked on the feature-direction’s associated words using the bag-of-words. Finally, this ranking is used to fine-tune the vector space and feature-directions so that the resulting feature-rankings are more faithful to the ranking on the bag-of-words.

This Chapter is a follow-up of the previous two Chapters, where previously feature-directions are identified in a variety of vector-spaces, and their potential applications are discussed, this Chapter focuses on improving the quality of these feature-directions to achieve better results. This Chapter continues with explaining the method to fine-tune a vector space and its associated feature directions using a bag-of-words in detail. Afterwards, we show quantitative results to see how the fine-tuning affects simple interpretable classifiers (as in Chapter ??). Finally, we end with a conclusion for potential future work.

## 6.2 Fine-Tuning Vector Spaces And Their Associated Feature Directions

To improve the directions and address these problems, we propose a method for fine-tuning the semantic space representations and corresponding feature directions. First, it is explained how to obtain target rankings from PPMI scores. Then, the neural network that uses these target rankings to improve the vector space and its associated feature-directions is described. The main idea is to use the BoW representations of the objects as a kind of weak supervision signal: if an object should be ranked highly for a given feature, we would expect the words describing that feature to appear frequently in its description. To obtain the target rankings, for each feature  $f$  we determine a total ordering  $\preceq_f$  such that  $o \preceq_f o'$  iff the feature  $f$  is more prominent in the BoW representation of object  $o'$  than in the BoW representation of  $o$ . We will refer to  $\preceq_f$  as the *target ranking* for feature  $f$ . If the feature directions are in perfect agreement with this target ranking, it would be the case that  $o \preceq o'$  iff  $v_C \cdot o \leq v_C \cdot o'$ . Since this will typically not be the case, we subsequently determine *target values* for the dot products  $v_C \cdot o$ . These target values represent the minimal way in which the dot products need to be changed to ensure that they respect the target ranking. Once these rankings have been obtained, we use a simple feedforward neural network to adapt the semantic space representations  $o$  and feature directions  $v_C$  to make the dot products  $v_C \cdot o$  as close as possible to these target values.



### 6.2.1 Generating Target Rankings

Let  $C_1, \dots, C_K$  be the clusters that were found using the method from Section 4.5.1. Each cluster  $C_i$  typically corresponds to a set of semantically related words  $\{w_1, \dots, w_n\}$ , which describe some salient feature from the considered domain. From the BoW representations of the objects, we can now define a ranking that reflects how strongly each object is related to the words from this cluster. To this end, we represent each object as a bag of clusters (BoC) and then compute PPMI scores over this representation. In particular, for a cluster  $C = \{w_1, \dots, w_m\}$ , we define  $n(C, o) = \sum_{i=1}^m n(w_i, o)$ . In other words,  $n(C, o)$  is the total number of occurrences of words from cluster  $C$  in BoW representation of  $o$ . We then write  $ppmi(C, o)$  for the PPMI score corresponding to this BoC representation, which is evaluated in the same way as  $ppmi(C, o)$ , but using the counts  $n(C, o)$  rather than  $n(w, o)$ . The target ranking for cluster  $C_i$  is then such that  $o_1$  is ranked higher than  $o_2$  iff  $ppmi(C_i, o_1) > ppmi(C_i, o_2)$ . By computing PPMI scores w.r.t. clusters of words, we alleviate problems with sparsity and synonymy, which in turn allows us to better estimate the intensity with which a given feature applies to the object. For instance, an object describing a violent movie might not actually mention the word ‘violent’, but would likely mention at least some of the words from the same cluster (e.g. ‘bloody’ ‘brutal’ ‘violence’ ‘gory’). Similarly, this approach allows us to avoid problems with ambiguous word usage; e.g. if a movie is said to contain ‘violent language’, it will not be identified as violent if other words related to this feature are rarely mentioned.

### 6.2.2 Generating Target Feature Values

Finding directions in a vector space that induce a set of given target rankings is computationally hard<sup>1</sup>. Therefore, rather than directly using the target rankings from Section 6.2.1 to fine-tune the semantic space, we will generate target values for the dot products  $v_{C_j} \cdot o_i$  from these target rankings. One straightforward approach would be to use the PPMI scores  $ppmi(C_j, o_i)$ . However these target values would be very different from the initial dot products, which among others means that too much of the similarity structure from the initial vector space would be lost. Instead, we will use isotonic regression to find target values  $\tau(C_j, o_i)$  for the dot product  $v_{C_j} \cdot o_i$ , which respect the ranking induced by the PPMI scores, but otherwise remain as close

<sup>1</sup>It is complete for the complexity class  $\exists\mathbb{R}$ , which sits between NP and PSPACE [?].

as possible to the initial dot products.

Let us consider a cluster  $C_j$  for which we want to determine the target feature values. Let  $o_{\sigma_1}, \dots, o_{\sigma_n}$  be an enumeration of the objects such that  $ppmi(C_j, o_{\sigma_i}) \leq ppmi(C_j, o_{\sigma_{i+1}})$  for  $i \in \{1, \dots, n-1\}$ . The corresponding target values  $\tau(C_j, o_i)$  are then obtained by solving the following optimization problem:

$$\textbf{Minimize:} \quad \sum_i (\tau(C_j, o_i) - v_{C_j} \cdot o_i)^2$$

**Subject to:**

$$\tau(C_j, o_{\sigma_1}) \leq \tau(C_j, o_{\sigma_2}) \leq \dots \leq \tau(C_j, o_{\sigma_n})$$

### 6.2.3 Fine-Tuning

We now use the target values  $\tau(C_j, o_i)$  to fine-tune the initial representations. To this end, we use a simple neural network architecture with one hidden layer. As inputs to the network, we use the initial vectors  $o_1, \dots, o_n \in \mathbb{R}^k$ . These are fed into a layer of dimension  $l$ :

$$h_i = f(Wo_i + b)$$

where  $W$  is an  $l \times k$  matrix,  $b \in \mathbb{R}^l$  is a bias term, and  $f$  is an activation function. After training the network, the vector  $h_i$  will correspond to the new representation of the  $i^{th}$  object. The vectors  $h_i$  are finally fed into an output layer containing one neuron for each cluster:

$$g_i = Dh_i$$

where  $D$  is a  $K \times l$  matrix. Note that by using a linear activation in the output layer, we can interpret the rows of the matrix  $D$  as the  $K$  feature directions, with the components of the vector  $g_i = (g_i^1, \dots, g_i^K)$  being the corresponding dot products. As the loss function for training the network, we use the squared error between the outputs  $g_i^j$  and the corresponding target values  $\tau(C_j, o_i)$ , i.e.:

$$\mathcal{L} = \sum_i \sum_j (g_i^j - \tau(C_j, o_i))^2$$

The effect of this fine-tuning step is illustrated in the right-most column of Table 6.1, where we can see that in each case the top ranked objects are now more closely related to the feature, despite being less common, and outliers such as ‘fence’ no longer appear.

20 Newsgroups	F1 D1	F1 D3	F1 DN
FT MDS	<b>0.50</b>	<b>0.47</b>	<b>0.44</b>
MDS	0.44	0.42	0.43
FT PCA	0.40	0.36	0.34
PCA	0.25	0.27	0.36
FT Doc2Vec	0.44	0.42	0.41
Doc2Vec	0.29	0.34	0.44
FT AWV	0.47	0.45	0.40
AWV	0.41	0.38	0.43
FT AWV <sub>w</sub>	0.41	0.41	0.43
AWV <sub>w</sub>	0.38	0.40	0.43
LDA	0.40	0.37	0.35

Table 6.2: Results for 20 Newsgroups.

## 6.3 Quantitative Evaluation

To evaluate our method, as in Chapter 6 we consider the problem of learning interpretable classifiers. In particular, we learn decision trees which are limited to depth 1 and 3, which use the rankings induced by the feature directions as input. This allows us to simultaneously assess to what extent the method can identify the right features and whether these features are modelled well using the learned directions. Note that depth 1 trees are only a single direction and a cut-off, so to perform well, the method needs to identify a highly relevant feature to the considered category. We can understand that the most demonstrable improvements for this method over the original directions will be in Depth 1 trees, as if the rankings for the important feature-directions are improved then these will be also. Depth 3 decision trees are able to model categories that can be characterized using at most three feature directions.

### Methodology

All tasks are evaluated as binary classification tasks. We randomly split the datasets into 2/3 for training and 1/3 for testing. For the place-types, we use 5-fold cross validation.

We used the logistic regression implementation from scikit-learn to find the directions.

Movie Reviews											
Genres	D1	D3	DN	Keywords	D1	D3	DN	Ratings	D1	D3	DN
FT MDS	<b>0.57</b>	<b>0.56</b>	0.51	FT MDS	<b>0.33</b>	<b>0.33</b>	0.24	FT MDS	<b>0.49</b>	<b>0.51</b>	<b>0.46</b>
MDS	0.40	0.49	<b>0.52</b>	MDS	0.31	0.32	<b>0.25</b>	MDS	0.46	0.49	<b>0.46</b>
FT AWV	0.42	0.42	0.39	FT AWV	0.25	0.25	0.15	FT AWV	0.47	0.44	0.39
AWV	0.35	0.44	0.43	AWV	0.26	0.21	0.19	AWV	0.44	0.48	0.41
LDA	0.52	0.51	0.45	LDA	0.22	0.19	0.18	LDA	0.48	0.48	0.41

Place-types											
Geonames	D1	D3	DN	Foursquare	D1	D3	DN	OpenCYC	D1	D3	DN
FT MDS	0.32	0.31	0.24	FT MDS	0.41	0.44	0.41	FT MDS	0.35	0.36	0.30
MDS	0.32	0.31	0.21	MDS	0.38	0.42	0.42	MDS	0.35	0.36	0.29
FT AWV	0.31	0.29	0.23	FT AWV	0.39	0.42	0.41	FT AWV	0.37	<b>0.37</b>	0.28
AWV	0.28	0.28	0.22	AWV	0.32	0.37	0.31	AWV	0.33	0.35	0.26
LDA	<b>0.34</b>	<b>0.32</b>	<b>0.27</b>	LDA	<b>0.55</b>	<b>0.48</b>	<b>0.47</b>	LDA	<b>0.40</b>	0.36	<b>0.31</b>

**Table 6.3: The results for Movie Reviews and Place-Types on depth-1, depth-3 and unbounded trees. .**

In Chapter 6 the hyper-parameters were chosen in stages. First, parameters for the best word-directions were found. Then, these best word-directions were taken and the best cluster parameters were found for these best word-directions. However, for these experimental results, we optimize the hyper-parameters together for word-directions, clustering and fine-tuning, where the best-parameters for each of these stages are those that ultimately produce the best-performing rankings for the fine-tuning on a decision tree. This is because fine-tuning is sensitive to which clusters and directions are included, as optimizing the ranking for one feature-direction may disrupt the ranking for another. This can be illustrated by the idea of optimizing a ranking for a direction on a noisy term like 'berardin', which refers to some metadata from the review text was optimized, then it's unlikely that this would benefit the other directions. However, if multiple directions that correspond to different genres were optimized like 'Horror' and 'Funny', then

IMDB Sentiment	D1	D3	DN
FT PCA	0.78	0.80	0.79
PCA	0.76	<b>0.82</b>	<b>0.80</b>
FT AWV	0.72	0.76	0.71
AWV	0.74	0.76	0.71
LDA	<b>0.79</b>	0.80	0.79

**Table 6.4: Results for IMDB Sentiment.**

it's likely that they would all benefit from a better representation. Cluster-directions are used because if all hyper-parameters are trained together, we can expect to find a set of directions that work with each other more easily than by limiting frequency for word-directions.

We evaluate for all domains described in Chapter 3 excluding Reuters. When learning word directions, only sufficiently frequent words are considered. In Chapter 6 this was chosen as a hyper-parameter, but as all parameters for each stage are tuned together it would take far too much time to optimize in this way, so it is chosen beforehand. It is chosen by pre-determining thresholds loosely based on the size of the vocabulary for the domain. We chose 100 for the movies dataset, 50 for the place-types, 30 for the 20 newsgroups dataset, and 50 for the IMDB sentiment dataset.

For hyperparameter tuning, we take 20% of the data from the training split as development data. We choose the hyperparameter values that maximize the F1 score on the development data for a Decision Tree on the improved feature-rankings that the fine-tuning network produces. As candidate values for the number of dimensions of the vector spaces we used  $\{50, 100, 200\}$ . The number of directions to be used as input to the clustering algorithm was chosen from  $\{500, 1000, 2000\}$ . The number of clusters was chosen from  $\{k, 2k\}$ , with  $k$  the chosen number of dimensions. For the hidden layer of the neural network, we fixed the number of dimensions as equal to the number of clusters. As the scoring metric for the dimensions, we considered accuracy, Kappa and NDCG. In all experiments, we used 300 epochs, a minibatch size of 200, and the tanh activation function for the hidden layer of the neural network. After some preliminary tests we found that in most cases the parameters for the network could be kept the same. In all experiments: 300 epochs, batch size 200 and tanh activation for the hidden layer. The hidden layer was kept the same size as the input space  $V_n$ . We train the network using AdaGrad [?],

with default values, and the model was implemented in the Keras library.

For the cluster size, we follow work by Steven Schockaert[?] and use twice the amount of clusters as there are dimensions in the space.

To learn the decision trees, we use the scikit-learn implementation of CART, which allows us to limit the depth of the trees. setting the maximum depth to one, three, or not at all. We used information gain as the attribute selection criterion. To mitigate the effects of class imbalance, the less frequent class was given a higher weight during training.

### 6.3.1 Results

Table 6.2 shows the results for the 20 newsgroups dataset, where we use FT to indicate the results with fine-tuning<sup>2</sup>. We can see that the fine-tuning method consistently improves the performance of the depth-1 and depth-3 trees, often in a very substantial way. After fine-tuning, the results are also consistently better than those of LDA. For the unbounded trees (DN), the differences are small and fine-tuning sometimes even makes the results worse. This can be explained by the fact that the fine-tuning method specializes the space towards the selected features, which means that some of the structure of the initial space will be distorted. Unbounded decision trees are far less sensitive to the quality of the directions, and can even perform reasonably on random directions. Interestingly, depth-1 trees achieved the best overall performance, with depth-3 trees and especially unbounded trees overfitting. Since MDS and AWW perform best, we have only considered these two representations (along with LDA) for the remaining datasets, except for the IMDB Sentiment dataset, which is too large for using MDS.

The results for the movies and place-types datasets are shown in Table 6.3. For the MDS representations, the fine-tuning method again consistently improved the results for D1 and D3 trees. For the AWW representations, the fine-tuning method was also effective in most cases, although there are a few exceptions. What is noticeable is that for movie genres, the improvement is substantial, which reflects the fact that genres are a salient property of movies. For example, the decision tree for the genre ‘Horror’ could use the feature direction for  $\{gore, gory, horror, gruesome\}$ .

<sup>2</sup>Since the main purpose of this first experiment was to see whether fine-tuning improved consistently across a broad set of representations, here we considered a slightly reduced pool of parameter values for hyperparameter tuning.

Some of the other datasets refer to more specialized properties, and the performance of our method then depends on whether it has identified features that relate to these properties. It can be expected that a supervised variant of this method would perform consistently better in such cases. After fine-tuning, the MDS based representation outperforms LDA on the movies dataset, but not for the place-types. This is a consequence of the fact that some of the place-type categories refer to very particular properties, such as geological phenomena, which may not be particularly dominant among the Flickr tags that were used to generate the spaces. In such cases, using a BoW based representation may be more suitable.

The results for IMDB Sentiment are shown in Table 6.4. In this case, the fine-tuning method fails to make meaningful improvements, and in some cases actually leads to worse results. This can be explained from the fact that the feature directions which were found for this space are themes and properties, rather than aspects of binary sentiment evaluation. The fine-tuning method aims to improve the representation of these properties, possibly at the expense of other aspects.

## 6.4 Conclusions

We have introduced a method to identify and model the salient features from a given domain as directions in a semantic space. Our method is based on the observation that there is a trade-off between accurately modelling similarity in a vector space, and faithfully modelling features as directions. In particular, we introduced a post-processing step, modifying the initial semantic space, which allows us to find higher-quality directions. We provided qualitative examples that illustrate the effect of this fine-tuning step, and quantitatively evaluated its performance in a number of different domains, and for different types of semantic space representations. We found that after fine-tuning, the feature directions model the objects in a more meaningful way. This was shown in terms of an improved performance of low-depth decision trees in natural categorization tasks. However, we also found that when the considered categories are too specialized, the fine-tuning method was less effective, and in some cases even led to a slight deterioration of the results. We speculate that performance could be improved for such categories by integrating domain knowledge into the fine-tuning method.

## **Appendix**

### **7.1 Chapter 3**

#### **7.1.1 Difference between Representations and Single Directions**



Newsgroups	D1				D2				D3			
	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec
PCA 200	0.254	0.097	0.373	-0.55	0.117	0.058	0.433	-0.41	0.004	0.099	0.134	0.078
PCA 100	0.259	0.135	0.345	-0.5	0.126	0.112	0.438	-0.367	0.006	0.157	0.118	<b>0.149</b>
PCA 50	0.277	0.133	0.277	-0.492	0.129	0.123	0.387	-0.347	0.006	0.177	0.228	0.143
AWV 200	0.145	0.133	0.1	-0.005	0.199	0.128	0.362	-0.414	0.194	0.185	0.441	-0.397
AWV 100	0.153	0.133	0.098	0.01	0.043	0.084	0.06	<b>0.109</b>	0.21	0.137	0.414	-0.474
AWV 50	0.11	0.122	0.088	0.044	0.056	0.088	0.068	0.052	0.21	0.142	0.362	-0.468
MDS 200	<b>0.378</b>	<b>0.234</b>	<b>0.439</b>	-0.498	<b>0.22</b>	0.203	0.509	-0.372	0.243	<b>0.257</b>	<b>0.568</b>	-0.406
MDS 100	0.271	0.178	0.138	-0.108	0.205	0.167	0.465	-0.401	<b>0.254</b>	0.217	0.506	-0.459
MDS 50	0.228	0.171	0.119	<b>0.115</b>	0.126	0.136	0.108	0.014	0.222	0.155	0.452	-0.476
D2V 200	0.149	0.138	0.101	0.037	0.158	0.202	<b>0.514</b>	-0.288	0.192	0.225	0.526	-0.365
D2V 100	0.162	0.166	0.123	0.041	0.169	<b>0.222</b>	0.474	-0.266	0.176	0.249	0.505	-0.306
D2V 50	0.162	0.181	0.132	0.08	0.154	0.193	0.452	-0.299	0.181	0.256	0.501	-0.314
Sentiment												
Reuters	D1	D2	D3	F1	ACC	F1	Prec	Rec	D1	D2	D3	F1
PCA	0.129	0.33	0.062	0.265	-0.002	-0.034	PCA	-0.006	0.053	0.042	0.044	0.032
AWV	<b>0.193</b>	0.345	0.008	0.327	<b>0.007</b>	<b>0.243</b>	AWV	0.057	0.047	0.068	0.042	0.018
MDS	0.184	<b>0.414</b>	0.08	0.349	0.003	0.231	D2V	<b>0.134</b>	<b>0.12</b>	<b>0.122</b>	<b>0.094</b>	<b>0.121</b>
D2V	0.159	0.316	<b>0.112</b>	<b>0.366</b>	0.006	0.188						
Movies												
Placetypes	D1	D2	D3	F1	ACC	F1	Prec	Rec	D1	D2	D3	F1
OpenCYC	ACC	F1	ACC	F1	ACC	F1	ACC	ACC	ACC	ACC	ACC	ACC
PCA	<b>0.047</b>	<b>0.025</b>	-0.003	0.038	<b>0.04</b>	<b>0.024</b>	PCA	0.102	0.111	<b>0.064</b>	0.101	<b>0.196</b>
AWV	0.036	0.021	<b>0.083</b>	<b>0.052</b>	0.027	-0.006	AWV	0.132	0.132	0.064	<b>0.115</b>	0.156
MDS	0.034	0.009	0.011	-0.012	0.016	-0.024	MDS	<b>0.17</b>	<b>0.148</b>	0.049	0.104	0.145
Keywords												
Foursquare	ACC	F1	ACC	F1	ACC	F1	ACC	ACC	F1	ACC	F1	ACC
PCA	0.054	0.135	<b>0.084</b>	<b>0.082</b>	0.008	<b>0.143</b>	PCA	0.09	0.04	<b>0.083</b>	0.034	0.022
AWV	<b>0.151</b>	<b>0.14</b>	0.053	0.02	<b>0.038</b>	0.014	AWV	<b>0.156</b>	0.041	0.031	0.024	<b>0.057</b>
MDS	-0.094	-0.022	0.075	0.055	0.038	0.031	MDS	0.111	<b>0.051</b>	0.05	<b>0.035</b>	0.023
Ratings												
Geonames	ACC	F1	ACC	F1	ACC	F1	ACC	ACC	F1	ACC	F1	ACC
PCA	<b>0.163</b>	0.047	0.063	0.037	<b>0.063</b>	0.011	PCA	-0.003	0.003	0.04	0.023	-0.003
AWV	0.054	<b>0.119</b>	0.04	<b>0.21</b>	-0.039	<b>0.106</b>	AWV	<b>0.045</b>	<b>0.041</b>	<b>0.074</b>	<b>0.042</b>	0.036
MDS	-0.035	-0.06	<b>0.078</b>	-0.007	-0.032	0.08	MDS	0.028	0.026	0.057	0.04	<b>0.045</b>

Table 7.1: The difference between the representations being directly input to the low-depth decision trees and the word directions

### **7.1.2 Class Names and Positive Occurrences**

Newsgroups	Positives	OpenCYC	Positives	FourSquare	Positives	Geonames	Positives	Genres	Positives	Ratings	Positives
alt.atheism	799	aqueduct	67	ArtsAndEntertainment	39	StreamLake	74	Action	2105	USA-G	1974
comp.graphics	973	border	556	CollegeAndUniversity	33	ParksArea	28	Adventure	1451	UK-12-12A	1566
comp.os.ms-windows.misc	985	building	91	Food	82	RoadRailroad	16	Animation	396	UK-15	3957
comp.sys.ibm.pc.hardware	982	dam	389	ProfessionalAndOtherPlaces	47	SpotBuildingFarm	176	Biography	627	UK-18	2009
comp.sys.mac.hardware	963	facility	173	NightlifeSpot	17	MountainHillRock	68	Comedy	4566	UK-PG	1724
comp.windows.x	988	foreground	43	ParksAndOutdoors	44	Undersea	27	Crime	2073	USA-PG-PG13	439
misc.forsale	975	historical_site	297	ShopsAndService	88	ForestHeath	14	Documentary	781	USA-R	5170
rec.autos	990	holy_site	44	TravelAndTransport	35			Drama	7269		
rec.motorcycles	996	landmark	96	Residence	6			Family	873		
rec.sport.baseball	994	medical_facility	28					Fantasy	928		
rec.sport.hockey	999	medical_school	49					Film-Noir	170		
sci.crypt	991	military_place	30					History	502		
sci.electronics	984	monsoon_forest	53					Horror	1963		
sci.med	990	national_monument	145					Music	1051		
sci.space	987	outdoor_location	103					Musical	529		
soc.religion.christian	997	rock_formation	184					Mystery	1128		
talk.politics.guns	910	room	60					Romance	2965		
talk.politics.mideast	940							Sci-Fi	1266		
talk.politics.misc	775							Short	560		
talk.religion.misc	628							Sport	385		
								Thriller	3293		
								War	671		
								Western	454		
Keywords (1)	Positives	Keywords (2)	Positives	Keywords (3)	Positives	Keywords (4)	Positives	Keywords (5)	Positives	Reuters	Positives
adultery	853	dancing	1655	funeral	802	money	887	shot-to-death	976	trade	466
bar	1334	death	2596	gore	820	mother-daughter-relationship	1477	singer	1278	grain	580
bare-breasts	1360	doctor	1193	gun	1445	mother-son-relationship	1908	singing	1372	nat-gas	105
bare-chested-male	1360	dog	1605	gunfight	776	murder	3496	song	986	crude	568
based-on-novel	2390	drink	1080	helicopter	864	new-york-city	1464	suicide	1092	sugar	162
beach	881	drinking	1246	hero	789	nudity	1887	surprise-ending	1202	corn	237
beating	1011	drunkmess	1291	horse	825	one-word-title	1357	tears	892	veg-oil	124
betrayal	848	escape	789	hospital	1434	party	1131	telephone-call	1187	ship	280
blood	2384	explosion	1283	hotel	902	photograph	1304	title-spoken-by-character	1725	coffee	139
boy	824	face-slap	907	husband-wife-relationship	2392	pistol	1378	topless-female-nudity	1079	wheat	283
boyfriend-girlfriend-relationship	1093	falling-from-height	875	independent-film	3431	police	1801	train	1069	gold	120
brother-brother-relationship	884	family-relationships	1787	infidelity	862	policeman	792	underwear	860	acq	2363
brother-sister-relationship	1025	father-daughter-relationship	1758	jealousy	928	pregnancy	821	violence	2231	interest	457
character-name-in-title	2146	father-son-relationship	2201	kidnapping	863	punched-in-the-face	870	voice-over-narration	1058	money-fx	676
chase	1351	female-nudity	2328	kiss	1759	rain	1053	watching-tv	887	soybean	111
church	897	fight	1356	knife	1097	restaurant	1202	wedding	800	oilseed	171
cigarette-smoking	1858	fire	1027	love	2164	revenge	1336	earn	3951	earn	3951
corpse	1008	fistfight	977	machine-gun	878	sequel	801	bop	104	bop	104
crying	1149	flashback	1937	male-nudity	1122	sex	2126	gnp	136	gnp	136
cult-film	1636	friend	1193	marriage	1407	shootout	1174	dlr	162	dlr	162
dancer	1020	friendship	1903	martial-arts	824	shot-in-the-chest	892	money-supply	168	money-supply	168

Table 7.2: Positive Instance Counts for each Class

# GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc.  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## 0. Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. Applicability and Definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format,  $\LaTeX$  input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. Copying in Quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. Modifications

you may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K.** For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M.** Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N.** Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties — for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known,



or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

## 6. Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. Aggregation with Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between

the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. Future Revisions of this License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.