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 .....

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

List of Publications

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

-
-

List of Figures

List of Tables

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^IMDB, 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^IMDB.

w

Corpus of Documents A unique group of words, e.g. a review from a domain of IMDB movie reviews DOM_IMDB.

C_dw

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 wf for its document D_wf.

wf

Bag-Of-Words a matrix BOW of documents BOW_D where each document is composed of unordered frequencies of words $D = [wf_1, ..., wf_n]$. 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_DC

Chapter 1

Introduction

Applications that enable user-generated content e.g. Social Media sites (Facebook, Twitter), Review sites (IMDB, Rotten Tomatoes, Amazon) and content-aggregation sites (Reddit, Tumblr) contain a deluge of text data e.g. in posts and comments. Research has shown methods using this data can solve a variety of problems, e.g. using text data to identify if social media posts, or product reviews, are positive or negative [**?**], to identify social media posts that happen during crises and identify those that are useful to crisis responders [**?**], or even to predict depression in social media users [**?**].

The methods to solve these problems are a result of machine-learning, which uses a large amount of data to learn how to solve some specific problem. Machine-learning is a force that has began to take-part in our day-to-day interactions with both the online and offline space, in the online world through content recommendation, targeted marketing and advertisement, and a variety of businesses that have sprung up with some machine-learning tool at the center. In the offline world, self-driving cars, face recognition on CCTV, behaviour prediction of crowds among others have started taking place across the world. Essentially, these tools are able to succeed because of the availability of this data.

However, concerns are growing about the difficulty of humans to interpret how these machine-learning models solve these problems.

Semantic spaces. 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 [33]. Conceptual spaces [8] 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 [11], 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).

1.1 Relationships

Vector spaces are representations that reduce the dimensionality of sparse representations like bag-of-words into dense spaces where semantic relationships e.g. two movies being similar to one another, are represented spatially. However, upon reducing this dimensionality the features are no longer interpretable. One way to interpret what these vector spaces mean follows Conceptual Spaces **??**, where entities in the domain e.g. movies in a domain of movie reviews are represented as points, and properties in the domain are represented as convex regions. The work in Chapter 5 details a process where the vector space is transformed so that these properties are used as features, creating an interpretable but dense representation. The introduction goes into further detail about these properties.

1.2 Contributions

1.3 Representations

1.4 Motivation

One task of Natural Language Processing is to obtain this semantic understanding from text by obtaining a machine-readable representation that contains domain knowledge. A basic approach

Entity: X

Word	Frequency
Dog	51
Cat	40
Man	11
Cheese	0

Dog	51
Cat	40
Man	11
Cheese	0

Entity: Y

Word	Frequency
Dog	51
Cat	40
Man	11
Cheese	0

Dog	51
Cat	40
Man	11
Cheese	0

Entity: Z

Word	Frequency
Dog	51
Cat	40
Man	11
Cheese	0

Dog	51
Cat	40
Man	11
Cheese	0

Figure 1.1: Bag-of-words

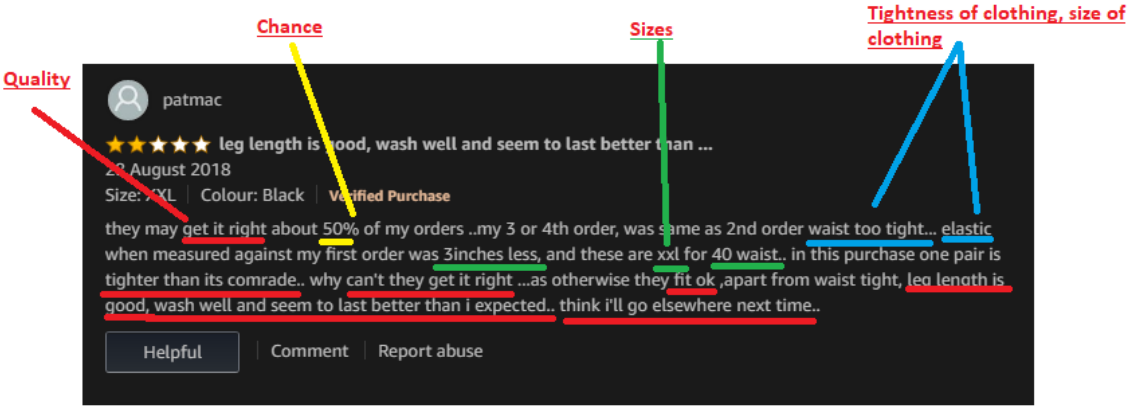

Figure 1.2: Example properties

to obtain a representation of this text is to represent entities (e.g. reviews, text-posts) by the frequency of their words, see 1.1.

Below, we show a review with its associated properties labelled.

1.4.1 Machine Learning

We can understand these properties to have a degree to which they apply, for example the size of the clothing might be "XXL", "XL", "L", "M" or "S", or the quality may be "Very good", "Good", "Ok", "Bad" or "Very bad". For the former, we may rely on the metadata available from the site itself, but for the latter the way to obtain this information is less clear. Although we may infer that the rating has some indication of these properties, it does not describe the properties or the degree to which the review refers to them. This kind of information is valuable for making sense of the world of unstructured text, and has broad applications, e.g. 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 [36]. For instance, [37] 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 property of being scary is adequately modelled as a direction in a semantic space of movies, such critiques can be addressed in a straightforward way. Similarly, in [19] a system was developed that can find “shoes like these but shinier”, based on a semantic space representation 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” [15]. While features such as popularity are typically not encoded in traditional knowledge bases, they can often be represented as semantic space directions.

1.4.2 Directions

However, manually labelling these properties and the degrees to which entities (e.g. reviews, text-posts) have them is extremely time-consuming.

A potentially ideal system would be as follows: We collect large amounts of unstructured text data, separated into domains, and obtain the properties of each domain from this data, and rank entities on the degree to which they have these properties. In this way, properties would be understood on a scale built from the domain directly, so that each domain has its own meanings for words according to their own idiosyncrasies. As the process does not require any manual labelling the quality of these properties could be improved simply by obtaining more data.

Further, as we are learning from unstructured data, not only would this allow us to understand the data in terms of what we know, but it would also introduce us to new ideas that we may not have previously understood. This kind of representation also has value in application to Machine Learning tasks. If we can separate the semantics of the space linearly into properties, we are able to learn simple linear classifiers that perform well.

Simple linear classifiers built from a representation composed of rankings on properties have an additional benefit of being more understandable.

1.5 Interpretability

Most successful approaches in recent times, like vector-spaces, word-vectors, and others, rely on the distributional model of semantics. This model relies on encoding unstructured text e.g. of a movie review, as a vector, where each dimension corresponds to how frequent each word is, we are able to calculate how similar the entities are, e.g. we know that if two movies have a similar distribution of words in their reviews, like frequent use of the word 'scary', or 'horror', then they would have a higher similarity value. These models, also known as 'semantic spaces' encode this similarity information spatially.

Semantic relationships can be obtained from semantic spaces.

applications/need for good interpretability:

- Safety
- Troubleshooting, bug fixing, model improvement
- Knowledge learning
- EU's "Right to explanation"
- Discrimination

properties of an interpretable classifier:

- Complexity: 'the magic number is seven plus or minus two' [30] also has many positive effects for its users, like lower response times [25, 13], better question answering and confidence for logical problem questions [13] and higher satisfaction [25].
- Transparancy:
- Explainability:
- Generalizability:

Properties, entities, the benefits and application of a representation formed of these

Basic introduction to directions, explanation of the utility and application of our approach

1.6 Thesis Overview / Contributions

In **??**, we focus on further experimenting with one relationship that was formalized in [6]: a ranking of entities on properties. In particular, we use this method of building a representation of entities as a way to convert a vector space into an interpretable representation, for use in an interpretable classifier. The reason that we chose this representation to expand on is because by representing each entity e with a vector v that corresponds to a ranking r, the meaning of each dimension is distinct, and we are able to find labels composed of clusters of words for these dimensions. Here, we make the distinction between a property and a word, a property is a natural property of the space that exists in terms of a ranking of entities, and words are the labels we use to describe this property.

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.

Similar to how it would not be possible for a human to solve a problem without a good understanding of the subject area, the first step of solving a problem with machine-learning is to obtain a suitable representation of the data. If this "understanding" or representation is not good, then no matter what steps are taken to try and solve the problem then they will not yield good results. 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 thesis refers 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 the 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 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 1.2 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 a representation from text data and using it to solve machine-learning problems, as well as giving a general introduction to related work. To outline the process, first as covered in Section 2.2.2 the data is preprocessed so that unnessecary information is removed. Then some basic representations are obtained in Section 2.3.1 and this is followed up by more complex vector space representations 2.4.6. To complete the process, 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 newsgroups 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 $r = (x_1, x_2, ..., x_n)$, 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 2.4.6.

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 unnessecary 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 $d = (wf_1, wf_2, ..., wf_n)$ 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 $r = (x_1, x_2, ..., x_n)$ 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 importantance to those that are frequent only in some documents. However, we are concerned with what distinguishes documents from each other so giving equal importantance 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 Freqency.

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

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:

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

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.

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

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. Below, we demonstrate this principal in a two-dimensional representation:

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 propogated 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 propogated 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 introuduced 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, ho 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 conceptual domain knowledge. For example, in a domain of movie reviews the "comedy" semantic relationship could be identified and used as a feature.

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 unanottated 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 [28] 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 Interpretablilty

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 interpretablily 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 concepts 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 illucidate 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 concept 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 concept 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 concepts 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

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 concepts 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 preceeded this one [6]. This work, instead of pursuing disentanglement as an objective, instead viewed vector spaces as "conceptual spaces". Conceptual spaces are a framework for vector space representation where entities are represented as points, and overlapping regions

that correspond to concepts in the domain encompass these entities. The factors of variation in this case were the features, or dimensions, of the vector space.

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 concepts in the domain. However, the degree to which they are factors of variation is not measured. Essentially, we view the representation as disentangled if the features obtained correspond to semantic features that are useful for 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 [3]. Beyond document representation, topic models have also been used to improve word embedding models, by learning a different vector for each topic-word combination [21].

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 Dirchlet 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, [**?**] [39] 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 concepts 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 the 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)

Chapter 3

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 <= 4 out of 10 or >= 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[1] 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

[1] 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[2], 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.

[2]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 [6]. 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 [6]. 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 [3] 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.

[3] 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[4] 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[5] Obtained from scikit-learn. [6] 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[7] 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%.

[4]Obtained by: https://keras.io/datasets/, Originally from https://ai.stanford.edu/ amaas/data/sentiment/ [22]

[5]http://qwone.com/ jason/20Newsgroups/

[6]https://scikit-learn.org/0.19/modules/generated/sklearn.datasets.fetch_20newsgroups.html#sklearn.datasets.fetch_20newsgroup

[7]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 [6]. 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): Following [5], we use Multi-Dimensional Scaling (MDS) to learn semantic spaces from the angular differences between the PPMI weighted BoW vectors.

This was chosen as it performed well in previous work by [6], and is a non-linear transformation based on PPMI vectors.

Principal Component Analysis (PCA): 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): Inspired by the Skipgram model [20]. A distributional document representation used as a representative of a higher performing method of learning in terms of document classification. For the Doc2Vec space, the hyper-parameters are additionally tuned for the $windowsize(5, 10, 15)$ referring to the context window, the $mincount(1, 5, 10)$ referring to the minimum frequency of words and the $epochs(50, 100, 200)$ of the network for each size space. We chose the best parameters for each class in each domain by evaluating the space as input to linear SVM's, which were tuned with two parameters: the best C values $1.0, 0.01, 0.001, 0.000]$ and if the weights should be balanced such that positive instances are weighted in proportion to how rare they are.

Average Word Vectors (AWV): Finally, we also use semantic spaces that are composed of averaged word vectors, using a pre-trained GloVe word embeddings trained on the Wikipedia 2014 + Gigaword 5 corpus[8]. While simply averaging word vectors may seem naive, this was found to be a competitive approach for unsupervised representations in several applications [12]. For each document, we simply average the vector representations of the words that appear at least twice in the BoW representation.

[8] `https://nlp.stanford.edu/projects/glove/`

Chapter 4

Re-organizing Vector Spaces into Interpretable Representations

4.1 Introduction

Vector space models encode meaning spatially, but their features are typically uninterpretable. This lack of interpretability limits their 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 [28], learning visual data alongside word-data to explain the content of images [23], and enforcing grammatical structure to perform better at question answering tasks [26].

This chapter is about re-structuring vector-space representations such that their features correspond to interpretable spatial structures in the original vector space. To give insight into what kind-of features this method can obtain, we can give an example from a domain 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[6], who first introduced the method to achieve this. The method begins with the following assumption: if a vector space can be linearly separated such that documents where a word occurs are separated from those where that word does not occur, that word is semantically important in the domain. This can be achieved in an unsupervised

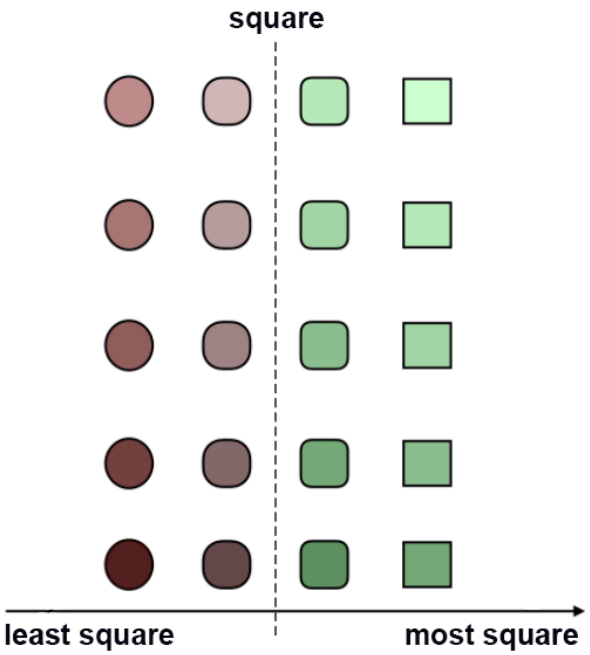

Figure 4.1: An example of a hyper-plane in a toy domain of shapes. The hyper-plane for the word square is the dotted line. Green shapes are positive examples and red shapes are negative examples. Those closest to the hyper-plane are less square than those further away.

way by training a linear model, e.g. a linear Support Vector Machine (SVM) (See Section **??**), to separate documents for words in a bag-of-words, where if a word occurs in a document it is treated as a positive example, and if it does not then it is a negative example. Then, 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 **??**).

In the case of a linear SVM, for each model trained trained on a word a hyper-plane is obtained separating 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. Given 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 by simply taking the orthogonal vector (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, we can obtain a ranking of that document on that word, e.g. how 'Scary' it is relative to the other documents. After repeating this process for all documents on a word direction, we can obtain a ranking of all documents on a word that can be used as a feature for the representation. This forms the basis of our approach towards restructuring a vector space such that the features encoded spatially are used directly as features of the representation. We first obtain hyper-planes for all words based on binary frequency. Then, we identify words which are semantically important (by e.g. F1 Score or accuracy of the hyper-plane). Finally, we obtain orthogonal directions for these word hyperplanes and rank documents on how far up they are on these directions. This ranking is then used as a feature of the new representation.

However, it can be sometimes unclear what a document being high up on a direction can mean when it's just a single word. 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 4.3). We show examples of these word cluster labels for the features of our interpretable representation in 4.1.

Directions in vector spaces 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 [35]. For instance, [38] 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 vector space of movies, such critiques can be addressed in a straightforward way. Similarly, in [18] a system was developed that can find "shoes like these but shinier", based on a vector space representation that was derived from

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.

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" [14]. While features such as popularity are typically not encoded in traditional knowledge bases, they can often be represented as semantic space directions. As another application, directions can also be used in interpretable classifiers. For example, [6] 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 2.4.9. For instance, [10] found that features of countries, such as their GDP, fertility rate or even level of CO_2 emissions, can be predicted from word embeddings using a linear regression model. In [17] directional vectors in word embeddings were found that correspond to adjectival scales (e.g. bad $<$ okay $<$ good $<$ excellent) while [29] found directions indicating lexical features such as the frequency of occurrence and polarity of words.

This work builds on the original method to find directions in a vector space and rank documents on them introduced by Derrac [6]. This chapter differs from that work by first introducing and explaining variants to the method in Section 4.4, then investigating directions qualitatively in Section **??**, examining how the new variants perform relative to each other, and finally an extensive quantitative examination in section **??** of all variants across all domains (as described in 4.1) to determine their usefulness in interpretable classifiers is explored. Finally, conclusions are made on the contribution of the chapter in section **??**. Chapter **??** builds on this method by applying and investigating its usage with vector spaces obtained from supervised neural networks,

and the Chapter **??** identifies problems with this method and introduces a novel unsupervised solution to improve performance.

4.2 Background

4.3 Clustering

4.3.1 K-means

4.3.2 Derrac's K-means Variation

4.4 Method

This section details the methodology to add structure to a vector space model starting with only itself and its associated Bag-Of-Words **??**. The work in this chapter differs from the method introduced by Derrac [6] as it focuses on achieving a new representation that can be applied in simple interpretable classifiers. Multiple variations are introduced and experimented on comprehensively.

word: w

document d

vector-space of documents V_d where $d = (x_1, x_2, ..., x_n)$ where x are features and $x(d)$ is equal to the value of a feature for a document

bag-of-words of documents B_d where $d = (wf_1, wf_2, ..., wf_n)$ and $wf(d)$ is equal to the frequency of that word in a document and n is equal to the number of unique words across all documents.

model for a word (M_w)

hyper-plane for a word = H_w

orthogonal direction vector = $\mathcal{D}_w$

ranking of all documents on a word direction $R_w = (rw_{d1}, rw_{d2}, ..., rw_{dn})$ where rw_d is equal to the ranking of a document on a word direction

cluster of words $C = (w_1, w_2, ..., w_n)$

cluster direction = $\mathcal{D}_C$

interpretable representation composed of rankings I_d where $d = (R_{w1}, R_{w2}, ..., R_{wn})$ and $R_w(d)$ is equal to the ranking of a document on a word direction

interpretable representation composed of cluster rankings I_d where $d = (R_{c1}, R_{c2}, ..., R_{cn})$ and $R_c(d)$ is equal to the ranking of a document on a cluster direction

ranking of documents on direction

4.4.1 Obtaining Directions and Rankings From Words

The method starts with a given vector-space 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 $d = (wf_1, wf_2, ..., wf_n)$ 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 $R_w = (rw_{d1}, rw_{d2}, ..., rw_{dn})$ 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 vector space model V_D. For this section, document vectors v_d are treated as points p_d in the space. For each word w, a hyper-plane h_w is obtained by training a linear model[1] M_w on the space V_D so that each document p_d in the space 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

[1]Tested using a logistic regression classifier and a linear SVM, both achieved similar results

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.[2]

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 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 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

[2]Using scikit-learn, class_weight:'balanced'

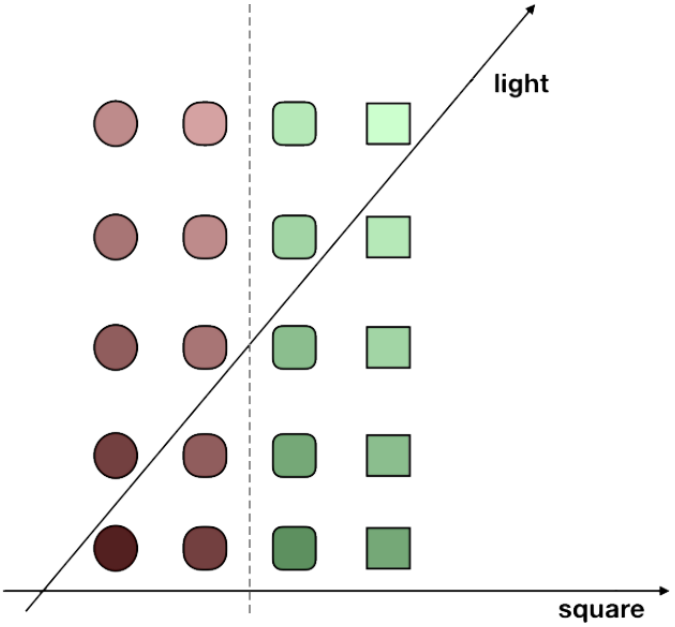

Figure 4.2: Another example of a hyper-plane in a toy domain of shapes. Here we show multiple directions, one for light and one for square The hyper-plane for the word square is the dotted line. Green shapes are positive examples and red shapes are negative examples for the word square. .

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[3]. 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.

4.4.2 Filtering Word-Directions

Although we are able to obtain word directions for every word, not every word is semantically important in the space. Here, we distinguish between word-directions $\mathcal{D}_w$ as directions that are

[3]The real cluster of words that this example comes from is "gore gory bloody blood gruesome serial investigate deaths"

not semantically important in the space, 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 [6] 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 Vector Space 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 space, as the detail of the Bag-Of-Words information is embedded in the space's structure.

After scoring the words using one of the aforementioned metrics, a simple cut-off is applied 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 $Fr = (fr_{d1}, fr_{d2}, ..., fr_{dn})$ 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 [6]. This metric evaluates the performance of the classifier, and also deals with the problem that these words are often

very imbalanced. In particular, 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 [16]. 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.

$$\text{DCG}_R^w = \sum_{i=1}^{pr_d} \frac{ppmi_i^w}{log_2(i+1)}$$
$$\text{IDCG}_R^w = \sum_{i=1}^{|documents|} \frac{2^{ppmi_i^w} - 1}{log_2(i+1)}$$
$$\text{nDCG}_R^w = \frac{\text{DCG}_R^w}{\text{IDCG}_R^w}$$

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.

4.4.3 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 interpretable classifier. The first is that there may be too many dimensions, so a classifier like a Decision Tree (See Section 2.4.1) 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" is a feature-direction, but what it represents can only be understood in the context of its cluster "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. There are two ways to solve this problem. The first is that the most similar directions (using cosine similarity) can be found and concatenated with the original word. However, this does not reduce the amount of features. By labelling feature-directions like this, we can rank documents on each of these directions to obtain associated labels for each feature that have context Fr_{wl} where $wl = (wl_{fr1}, wl_{fr2}, ..., wl_{frn})$ and wl_{fr} is a group of words to label a feature.

The second is that the directions themselves can be clustered, such that feature-directions are clustered together to obtain new cluster features $cr = C(w_1, w_2, ..., w_n)$ where C is the clustering algorithm and w are words that it has chosen to cluster together. From these new cluster

features, a new representation I_{Cr} is taken composed of rankings on feature cluster-directions $Cr_i = (cr_{d1}, cr_{d2}, ..., cr_{dn})$ where cr_d is the ranking of a document on a cluster feature-direction. Associated labels are obtained by simply concatenating the words of the feature-directions that are clustered together These clustered feature-directions can be obtained for example by averaging all feature-directions that are clustered together. 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 concepts 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 concept of a horror movie more accurately.

On the other hand, its possible that when clustering many words together the cluster feature-direction no longer represents a semantically important feature. For example given the associated label for a cluster feature-direction $\{Romance, Love\}$ and a cluster feature-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 for animal movies - but a balance must be struck between finding the semantically important clusters in the space and creating new clusters that may not be as semantically important because that word does not easily fit into a cluster. In the quantitative results, sometimes clustering performed worse than single directions, and not being able to find this balance for the specific classes in question can be attributed as to why, specifically because clusters were not semantically important enough or were disrupted by clustering together words that do not

fit.

We will experiment with two different clustering methods: k-means and a variant of k-means that was proposed in the work by Derrac [6]:

K-Means In the experimental results, it was found that Derrac's variation relies too much on its initial directions, meaning if a noisy direction is chosen as the first cluster centre, then key directions may be missed. Avoiding this is difficult without extensive and sometimes arbitrary hyper-parameter optimization. For this reason, it was decided to also consider K-Means as an alternative clustering algorithm. K-means traditionally begins with K centroids c randomly placed into the space. In our case, these centers are weighted according to the squared distance from the closest center already chosen. [1] Then, the distance between each point d_p and centroid c is calculated. In-order for Euclidean distance to be meaningful, directions are normalized making Euclidean distance the same as cosine similarity. Each point p is then assigned to its closest centroid c. Then, the centroids are recomputed to be the mean of their assigned points. This process starting with the distance calculation is repeated until the points assigned to the centroids do not change.

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, where N is a hyperparameter. 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 step is a set of clusters $C_1, ..., C_K$, where each cluster C_j is identified with a set of words. In the following, we will write v_{C_j} will be written to denote the centroid of the directions corresponding to the words in the cluster C_j, which can be computed as $v_{C_j} = \frac{1}{|C_j|} \sum_{w_l \in C_j} \frac{v_l}{\|v_l\|}$ provided that the vectors v_w are all normalized.

The first cluster centroid is chosen by taking the top-scoring direction for a scoring metric. Then, centroids are selected until the desired number is reached by taking the maximum of the summed absolute cosine similarity of all current centroids, in other words taking the most dissimilar direction to all of the current directions. Once the centroids are selected, for each remaining direction the centroid is found it is most similar to, and the centroid is updated once the direction has been added.

Meaning that the key is to rank documents on the initial direction only, and only use the remaining features in each cluster to provide a more informative label if the clusters are too noisy.

4.5 Qualitative Results

In principle, NDCG should be better suited for gradual features. For example, a binary feature would be 'Gore', where a film is either gory or not gory. A gradual feature would be "rating", referring to the age rating for films and gradually increasing. In practice, however, there was not such a clear pattern in the differences between the words chosen by these metrics despite often finding different words. Put another way, it is difficult to say if the words highly scored by NDCG are more gradual than other scoring metrics.

4.5.1 Datasets

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 all of these datasets, we split them into a 2/3 training data, 1/3 test data split. We additionally remove the end 20% of the training data and use that as development data for our hyper-parameters, which is then not used for the final models verified using test data. For the movies and place-type domains, the original text was not available.

4.5.2 Space Types

Below the choices for the Vector Space Models that are formally described in Section **??** are explained:

Multi-Dimensional Scaling (MDS): Following [5], we use Multi-Dimensional Scaling (MDS) to learn semantic spaces from the angular differences between the PPMI weighted BoW vectors.

Principal Component Analysis (PCA): directly uses the PPMI weighted BoW vectors as input, and which avoids the quadratic complexity of the MDS method. A standard dimensionality reduction technique, used as a baseline reference.

Doc2Vec (D2V): Inspired by the Skipgram model [20]. A distributional document representation used as a representative of a higher performing method of learning in terms of document classification. For the Doc2Vec space, the hyper-parameters are additionally tuned for the $windowsize(5, 10, 15)$ referring to the context window, the $mincount(1, 5, 10)$ referring to the minimum frequency of words and the $epochs(50, 100, 200)$ of the network for each size space. The process with our two-part hyperparameter optimization as in this case is as follows: Grid search is used to select the parameters for the representation, then find the most suitable model (e.g. Decision Tree, SVM) for that representation.

Average Word Vectors (AWV): Finally, we also learn semantic spaces by averaging word vectors, using a pre-trained GloVe word embeddings trained on the Wikipedia 2014 + Gigaword 5 corpus[4]. While simply averaging word vectors may seem naive, this was found to be a competitive approach for unsupervised representations in several applications [12]. We simply average the vector representations of the words that appear at least twice in the BoW representation.

4.5.3 The best-performing 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. These are arranged from highest scoring to least scoring, with the score-type and space-type chosen by performance. These are not clusters, but rather single directions with the two most similar directions in brackets beside them for context. This is the alternative way of presenting these directions as mentioned at the start of Section 4.4.3.

There is an interesting difference between the sentiment directions and the movies directions in the examples below. 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 concepts 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. This is likely because the PPMI scores for actor names would be high as they are both rare and definitive for movies. For the newsgroups domain, a number of directions are seen that are likely to only belong to a certain newsgroups, e.g. you would find the word 'celestial' more often in the religious sections than the others, and the

[4] `https://nlp.stanford.edu/projects/glove/`

word 'diesel' more often in the automobile section but not others. This is an expected natural clustering of the domain into its 20 newsgroups. The place-types section generally describes either aspects of the camera (e.g. canon60d), aspects of the photo (greyscale) or features found in the photo (gardening). The former likely relates to the degree to which filters or editing has been applied to the photo, while the latter makes more sense for our classification task. For the reuters dataset, the highest scored semantics seem to generally be related to dates (1st, may, june), however there is also some business jargon (quarterly, avg, dlr).

Movies *(50 MDS NDCG)*	**Sentiment** *(100 D2V NDCG)*	**Newsgroups** *(50 D2V NDCG)*	**Place-types** *(50 PCA Kappa)*	**Reuters** *(200 MDS NDCG)*
horror (scares, scary)	glenda (glen, matthau)	karabag (iranian, turkiye)	blackcountry (listed, westmidlands)	franklin (fund, mthly)
hilarious (funniest, hilarity)	scarlett (gable, dalton)	leftover (flaming, vancouver)	ears (stare, adorable)	quarterly (shearson, basis)
bollywood (hindi, india)	giallo (argento, fulci)	wk (5173552178, 18084tmibmclmsuedu)	spagna (espanha, colores)	feb (28, splits)
laughs (funnier, funniest)	bourne (damon, cusack)	1069 (mlud, wibbled)	oldfashioned (winery, antiques)	22 (booked, hong)
jokes (gags, laughs)	piper (omen, knightley)	providence (norris, ahl)	gardening (greenhouse, petals)	april (monthly, average)
comedies (comedic, laughs)	casper (dolph, damme)	celestial (interplanetary, bible)	pagoda (hindu, carved)	sets (principally, precious)
hindi (bollywood, india)	norris (chuck, rangers)	mlud (wibbled, 1069)	artificial (saturation, cs4)	16 (creditor, trillion)
war (military, army)	holmes (sherlock, rathbone)	endif (olwm, ciphertext)	inner (curved, rooftops)	1st (qtr, pennsylvania)
western (outlaw, unforgiven)	rourke (mickey, walken)	gd3004 (35894, intergraph)	celebrate (festive, celebrity)	26 (approve, inadequate)
romantic (romance, chemistry)	ustinov (warden, cassavetes)	rtfmmitedu (newsanswers, ieee)	vietnamese (ethnic, hindu)	23 (offsetting, weekly)
songs (song, tunes)	scooby (doo, garfield)	eng (padres, makefile)	cn (elevated, amtrak)	prior (recapitalization, payment)
sci (science, outer)	doo (scooby, garfield)	pizza (bait, wiretap)	mannequin (bags, jewelry)	avg (shrs, shr)
funniest (hilarious, funnier)	heston (charlton, palance)	porsche (nanao, mercedes)	falcon (r, 22)	june (july, venice)
noir (noirs, bogart)	homer (pacino, macy)	gebcadredslpittedu (n3jxp, skepticism)	jewish (monuments, cobblestone)	march (31, day)
documentary (documentaries, footage)	welles (orson, kane)	scsi2 (scsi, cooling)	canon60d (kitlens, 600d)	regular (diesel, petrol)
animation (animated, animators)	frost (snowman, damme)	playback (quicktime, xmotif)	reflective (curved, cropped)	4th (qtr, fourth)
adults (adult, children)	streisand (bridget, salman)	35894 (gd3004, medin)	mason (edward, will)	27 (chemlawn, theyre)
creepy (spooky, scary)	davies (rhys, marion)	diesel (volvo, shotguns)	aerialview (manmade, largest)	14 (borrowing, borrowings)
gay (gays, homosexuality)	cinderella (fairy, stepmother)	evolutionary (shifting, hulk)	shelf (rack, boxes)	11 (chapter, ranged)
workout (intermediate, instruction)	boll (uwe, belushi)	techniciandr (obp, 144k)	monroe (raleigh, jefferson)	may (probably, however)
thriller (thrillers, suspense)	rochester (eyre, dalton)	8177 (obp, 144k)	litter (fujichrome, e6)	38 (33, strong)
funnier (laughs, funniest)	edie (soprano, vertigo)	shaw (medicine, ottoman)	streetlights (streetlamp, headlights)	m1 (m2, m3)
suspense (suspenseful, thrillers)	scarecrow (zombies, reese)	scorer (gilmour, lindros)	carlzeiss (f2, voigtlander)	dlr (writedown, debt)
arts (hong, chan)	kramer (streep, meryl)	xwd (xloadimage, openwindows)	manmade (aerialview, below)	five (years, jones)
christianity (religious, religion)	marty (amitabh, goldie)	ee (275, xloadimage)	demolished (neglected, rundown)	bushels (soybeans, ccc)
musical (singing, sing)	columbo (falk, garfield)	com2 (com1, v32bis)	wald (berge, wildflower)	revs (net, 3for2)
gore (gory, blood)	kidman (nicole, jude)	examiner (corpses, brass)	arquitetura (exposition, cidade)	29 (175, include)
animated (animation, cartoon)	juliet (romeo, troma)	migraine (ama, placebo)	greyscale (highcontrast, monochromatic)	acquisition (make, usairs)
gags (jokes, slapstick)	garland (judy, lily)	parliament (parliamentary, armored)	alameda (monday, marin)	payable (div, close)

Table 4.2: The top-scoring words for each domain, scoring metric and space type determined by the highest F1-score

4.5.4 Comparing Space Types

To select these quantitative examples for comparing score types, it was first demonstrated on the movies domain to be consistent with previous examples. However, as this does not contain the doc2vec space, additional results are provided in the next section for the newsgroups. The space that performed well on the genres task for the movies is used, with the understanding that genres as a key natural classification task will likely give good example directions that correspond to domain knowledge. After selecting this space, the same sized spaces are chosen from the other space-types (size 200). The same score-type and frequency cut-off as the best performing space-type are also used. In this case, the best performing type for the PCA space was 20,000 frequency cutoff and NDCG. So even though sometimes a different frequency cut-off performed better for the other space-types, this is equalized so that the words are the same. This means that sometimes the space-type is a slightly worse performing one than chosen as the final results, and that the original space has a performance advantage, but this makes the results more consistent. These qualitative experiments are approached with the following idea: spaces that perform better on natural domain tasks using Decision Trees contain unique natural directions that other spaces do not have.

The commonalities between spaces are much more prevalent than the differences, with natural concepts of the domain being represented in all of the different space types. However, different spaces do perform better than others on natural domain tasks. For this reason, the directions which are unique to each space-type are shown.

When examining the table of results, it can be observed that the common terms are mostly salient concepts relevant to the domain. However, MDS has the most unique general concepts relevant to the domain that others do not have. AWV contains names, and concepts which are interesting but more related to specific aspects than genre (train, slaves). Meanwhile PCA seems to prioritize words in the reviews that are not concepts but rather parts of sentences (surprisingly, admit, talents, tired, anymore). However, both PCA and MDS contain unique noisy terms as well. The term 'berardinelli' and 'rhodes' for MDS as well as 'compuserve' for PCA are artifacts of the data being obtained from the web. Despite this, it seems that MDS does contain more interesting unique directions than PCA, and as it performed best on the genres task, this makes sense.

MDS	AWV	PCA	Common
berardinelli *(employers, distributor)*	billy *(thrown, dirty)*	amount *(leaving, pick)*	noir *(fatale, femme)*
crawford *(joan, davis)*	brother *(brothers, boys)*	fails *(fit, pick)*	gay *(homosexual, homosexuality)*
hitchcocks *(hitchcock, alfred)*	fonda *(henry, jane)*	pick *(fails, fit)*	prison *(jail, prisoners)*
warners *(warner, bros)*	building *(built, climax)*	stands *(fails, cover)*	arts *(rec, robomod)*
nuclear *(weapons, soviet)*	train *(tracks, thrown)*	surprisingly *(offer, fit)*	allens *(woody, allen)*
joan *(crawford, barbara)*	slaves *(slavery, excuse)*	copyright *(email, compuserve)*	jokes *(laughs, joke)*
kidnapped *(kidnapping, torture)*		length *(reflect, expressed)*	animation *(animated, cartoon)*
hop *(hip, rap)*		profanity *(reflect, producers)*	sherlock *(holmes, detective)*
kung *(martial, jackie)*		compuserve *(copyright, internetreviews)*	western *(westerns, wayne)*
ballet *(dancers, dancer)*		talents *(admit, agree)*	songs *(song, lyrics)*
gambling *(vegas, las)*		admit *(agree, talents)*	comedies *(comedic, laughs)*
alcoholic *(drunk, alcoholism)*		developed *(introduced, sounds)*	workout *(exercise, challenging)*
waves *(surfing, wave)*		intended *(bother, werent)*	laughs *(funnier, hilarious)*
jaws *(jurassic, godfather)*		constantly *(putting, sounds)*	drug *(drugs, addict)*
jungle *(natives, island)*		tired *(anymore, mediocre)*	sci *(science, fiction)*
employers *(berardinelli, distributor)*		produced *(spoiler, surprising)*	documentary *(documentaries, interviews)*
pot *(weed, stoned)*		involving *(believes, belief)*	students *(student, schools)*
canadian *(invasion, cheap)*		anymore *(continue, tired)*	thriller *(thrillers, suspense)*
murphy *(eddie, comedian)*		leaving *(fit, pick)*	allen *(woody, allens)*
comics *(comedian, comedians)*		makers *(producers, aspects)*	funniest *(hilarious, laughing)*
kidnapping *(kidnapped, torture)*		introduced *(developed, considered)*	gags *(jokes, slapstick)*
subscribe *(email, internetreviews)*		loses *(climax, suffers)*	adults *(children, adult)*
vegas *(las, gambling)*		negative *(positive, bother)*	animated *(animation, cartoon)*
distributor *(berardinelli, employers)*		expressed *(reflect, opinions)*	dancing *(dance, dances)*
wave *(waves, surfing)*		mildly *(mediocre, forgettable)*	teen *(teenage, teens)*
rhodes *(internetreviews, email)*		helped *(putting, allowed)*	soldiers *(soldier, army)*
hippie *(pot, sixties)*		reflect *(expressed, opinions)*	indie *(independent, festival)*
weed *(pot, stoned)*		opinions *(reflect, expressed)*	suspense *(suspenseful, thriller)*
caribbean *(pirates, island)*		frequently *(occasionally, consistently)*	creepy *(scary, eerie)*
eddie *(murphy, comedian)*		content *(agree, proves)*	italian *(italy, spaghetti)*
sixties *(beatles, hippie)*		allowed *(helped, werent)*	jews *(jewish, nazis)*
... 8 More		suffers *(lacks, loses)*	... 1480 more

Table 4.3: Unique terms between space-types

Score Types

There are unique directions for each different space type from the movies domain, each suitable to different tasks. Obtained in the same way as before, this time the 200 MDS space is used that performed the best on the genres task and found those unique to it. Once again, the most understandable and general concepts are those that are common to all score-types. NDCG performed the best on most tasks, and it can be seen that a lot of new concepts are introduced in NDCG compared to the other scoring types. F1 by and large seems is difficult to understand, referring to names or specific aspects of the scene, and accuracy is similar. Kappa has some unique sentiment related terms, as as well as some aspects of the presentation of the film (featurette, critic, technical), but it does not contain unique general concepts the way NDCG does. It can be surmised that as NDCG contains these unique conceptual directions, it is able to perform better than other score-types.

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

Table 4.4: Different score types

Comparing PPMI representations to doc2vec

Now in Table a comparison is shown between a time when doc2vec was the highest performing representation, in this case on the newsgroups domain. Doc2vec is compared to MDS in this case as MDS also performed well. This is to see if doc2vec, by making use of word-vectors and word-context can find interesting unique directions compared to MDS, which was obtained from a PPMI BOW. In general, it is found that MDS contains a lot more irrelevant words than D2V, specifically related to parts-of-words. It seems that doc2vec was better at recognizing these words as noise and uninteresting compared to PPMI, which must have prioritized these words. Doc2Vec also brings up some interesting concepts, e.g. cryptology, which is very relevant to the 20 newsgroup subtype of cryptography. 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.

4.6 Quantitative Results

For all of these datasets, we split them into a 2/3 training data, 1/3 test data split. We additionally remove the end 20% of the training data and use that as development data for our hyper-parameters, which is then not used for the final models verified using test data.

4.6.1 Evaluation Method

Primarily the effectiveness of a representation is evaluated on its ability to perform in low-depth Decision Trees, specifically CART Decision Trees (See Background Section 2.4.1) with a limited depth of one, two and three. This evaluation has a few assumptions: A good interpretable representation disentangles salient domain knowledge into its dimensions, and natural domain tasks (e.g. classifying genres of movies using their reviews) can be evaluated effectively using that salient domain knowledge. Put another way, if the space is representing domain knowledge well it can be expected that the space is linearly separable for key semantics of the domain. In spatial terms, a representation will be capable of being linearly transformed by our method

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)	bobbeviceicotekcom (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, bobbeviceicotekcom)
5173552178 (circumference, wk)	mail (send, list)	would (appreciate, wouldnt)
18084tmibmclmsuedu (circumference, wk)	doesnt (isnt, mean)	beauchaine (bobbeviceicotekcom, 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 sapce to a D2V space for Newsgroups, where a D2V space performed best..

4.5.4

into these distinct relevant concepts if semantically distinct entities are spatially separated, and semantically similar entities are close together.

If only the the quality of the representation was being evaluated, only Linear SVM's could be used to find the hyper-planes that effectively separate these spatial representations for the class. However, the representations that encode this spatial information are not interpretable, so a linear classifier although able to separate the documents that contain the class and do not contain them will not be interpretable either. It is our main interest to evaluate how well a representation encodes these key semantics while also being restricted by the requirement to be

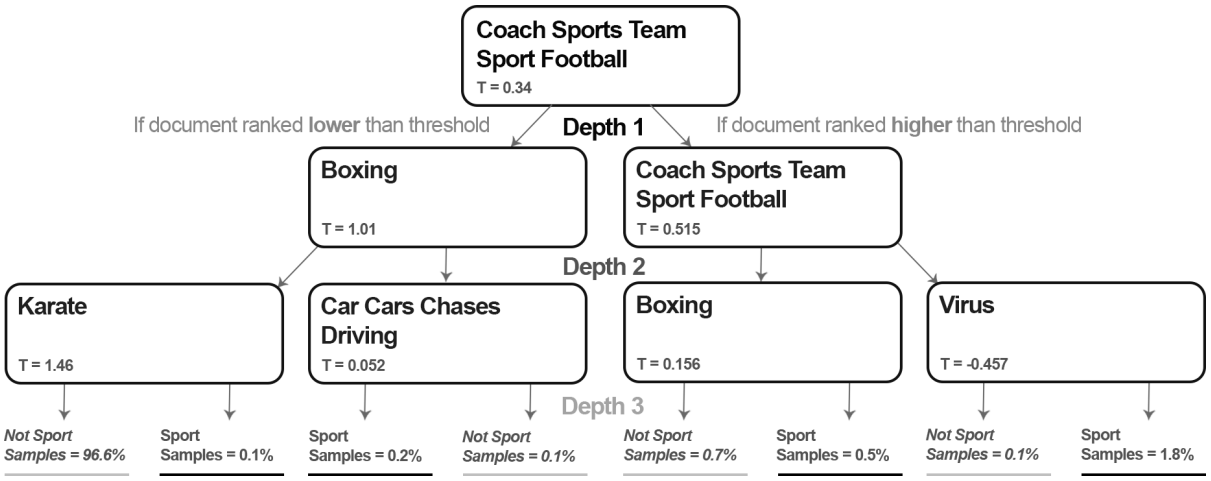

Figure 4.3: An example of a Decision Tree classifying if a movie is in the "Sports" genre. Each Decision Tree Node corresponds to a feature, and the threshold T is the required ranking of a document on that feature to traverse right down the tree instead of left. One interesting point to note is that 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) .

disentangled into words or clusters, in other words how well it represents the information while also being interpretable.

Given these assumptions, low-depth Decision Trees can give an estimation of how good an interpretable representation is. If the representation cannot perform for a class at a one-depth tree, then it is not disentangled such that it contains a single salient dimension that effectively evaluates a class. If a representation cannot perform well on two-depth trees, then the representation is not disentangled into three concepts that can sufficiently determine that class, and if a representation cannot perform well on three-depth trees, it has not disentangled the representation such that there are nine relevant concepts that are relevant to that class. To see what these different trees look like see Figure **??**. A comparison to put this in better perspective is to an unbounded tree. Unbounded trees select a large amount of dimensions in order to achieve a performance difference on development data, but when applied to test data the models do not generalize well. This is because they overfit, rather than using the key semantics of the space to classify.

Primarily F1-score is used to determine if a classifier is good or not. This is because many of the classes are unbalanced so accuracy is not a good metric, as high accuracy could be achieved by predicting only zeros. All of the results shown in this section are the end-product of a two-part hyper-parameter optimization. Each Decision Tree has its own set of hyper-parameters that are optimized as does each representation-type. These are the models trained on the training data and scored on the test data, with the highest performing in terms of F1-score parameters from hyper-parameter optimization on the development data. For ease of comparison, some results are provided with SVM's and unbounded Decision Trees, as well as a baseline Topic Model, which is used as a reference for a standard interpretable representation. Below, the parameters are listed that are optimized for each of these model types:

Linear Support Vector Machines (SVM's)??: C parameters and gamma parameters. C 1.0, 0.01, 0.001, 0.0001, Gamma 1.0, 0.01, 0.001, 0.0001.

Topic Models??: Two priors: The doc topic prior 0.001, 0.01, 0.1 and the topic word prior 0.001, 0.01, 0.1

CART Decision Trees 2.4.1: The number of features to consider when looking for the best split. $None, auto, log2$ and the criterion for a node split $criterion : gini, entropy$.

For the baselines, four different Vector Space Models are used, a Bag-Of-Words of PPMI (BOW-PPMI) scores and a standard Latent Dirchlet Allocation (LDA) Topic Model. As well as the original filtering done to the representations, for the BOW-PPMI additionally all terms are filtered out that do not occur in at least $(d_N/1000)$ documents. Otherwise, there would be too many irrelevant terms to be a fair comparison. The dimension amounts that are compared are of size $(50, 100, 200)$. The MDS space is not available for sentiment, as the memory cost was too prohibitive with 50,000 documents, and there are no doc2vec spaces for placetypes/movies, as it was only possible access to the Bag-Of-Words representation.

When obtaining the single word directions, starting with all of the baseline representations and vocabularies, the infrequent terms are filtered from these vocabularies according to a hyper-parameter that is tuned. As the doc2vec has already been hyper-parameter optimized, the optimal doc2vec space that scored the highest for its class on a Linear SVM is used, rather than tuning the entire process around the doc2vecs vectors. So for example, when evaluating the Keywords task for the movies, directions are obtained from the doc2vec space that performed

best for a linear SVM on the Keywords task following the previous experiments.

Results are obtained for the rankings induced from these word directions on Decision Tree's limited to a depth-three in-order to select the best parameters when using directions for each class. The parameters that are desirable to determine are the type of Vector Space Model, the size of the space, the frequency threshold and the score threshold, which determines the top scoring directions. To do so, for each space-type of each size, a grid search is used to find the best frequency and score cut-offs for that sized space-type. Then, from these space-types and sizes the best performing one is selected. There is a balance between finding words which are useful for creating salient features in our clustering step without including too many words which do not. As our clustering methods are unsupervised, it is important that the amount of noise being entered into them is limited, despite the classifiers that use these directions typically being able to filter out those directions which are not suitable to the class. Additionally, as the vocabulary size varies from dataset to dataset, the threshold will naturally be different for each one.

These results allow us to choose for each class, the best Vector Space Model and size of that model for that class. We obtain directions and rankings for all different space types and sizes. Next, we test single directions, attempting to find a good amount of directions to cluster and not including words which may hamper the unsupervised classification, as well as the best space-type for each domain. We found that generally, classifiers performed better with more data, so we use 20000 as our frequency cutoff and 2000 as our score cutoff. Our hyper-parameters for the frequency cut-off were 5000, 10000 and 20000, and our hyper-parameters for the score-cutoff were 1000 and 2000.

For the clustering, the optimal directions as decided by the single direction experiments is chosen, and the same frequency and score thresholds are used as was optimally chosen. Two different clustering algorithms are experimented with: Derrac and K-Means. As these algorithms select centroids from the top-scoring directions or randomly, we can expect that some clusters may not be salient features of the space. This is because top-scoring directions, e.g. for accuracy could simply infrequent terms that do not have much meaning, and these infrequent terms could also be randomly selected. We could use grid-search on the frequency and score cutoffs when obtaining these results in order to avoid terms that may disrupt existing clusters or form cluster centers that are not salient features of the space, but we chose a more standardized process that

would rely on the parameters of the clustering algorithms and the ability of the classifiers to filter out clusters that are not informative, so as to not make a time-costly grid search a necessary part of the process.

For K-means clustering, we use Mini batch K-means, implemented by scikit-learn [5], introduced by [32] and kmeans++ to initialize [1]

4.6.2 Summary of all Results

To begin, the original dimensions of the space are compared to the rankings on single words, the rankings on cluster directions, and the Bag-Of-Words of PPMI scores and topic models on low-depth Decision Trees.

In general, all spaces that are not transformed do not perform well on this task. We hypothesize that this is because their dense dimensions are not semantically independently. In contrast, single-directions and clusters of these single-directions obtained from these spaces out-perform the bag-of-words in most cases, with the exceptions being in the place-types domain and the keywords task for the movies.

For the keywords task, the natural explanation is that in a depth-1 tree, finding words which are directly corresponding to particular keywords is easier with words than if using directions, not only because certain words may have been filtered out, but also because as they are infrequent they may not be well-represented in the space. In this case, the PPMI representation is perfect, as it can find 1-1 matches with the classes without the representations of those words being spatially influenced by other similar words, as it can be expected for them to be in the space. However, this changes when going from depth-one to depth-two and depth-three, which is likely due to overfitting in the case of the PPMI representation. 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 is our most unbalanced domain with the least documents, so it is possible that they overfit.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

	Genres			Keywords			Ratings		
Movies	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	**0.436**	0.463	0.492	0.23	**0.233**	**0.224**	0.466	0.499	0.498
Clusters	0.431	**0.513**	**0.506**	0.215	0.22	0.219	**0.504**	**0.507**	**0.513**
PPMI	0.429	0.443	0.483	**0.243**	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		
	D1	D2	D3	D1	D2	D3	D1	D2	D3
Rep	0.251	0.366	0.356	0.705	0.77	0.773	0.328	0.413	0.501
Single dir	0.418	**0.49**	**0.537**	0.784	0.814	**0.821**	**0.678**	**0.706**	0.72
Cluster	0.394	0.433	0.513	0.735	**0.844**	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	**0.723**
Topic	**0.431**	0.423	0.444	**0.79**	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	**0.541**	0.498	**0.531**	0.404	**0.428**	0.39	**0.444**	**0.533**	**0.473**
Cluster	0.462	0.507	0.496	**0.413**	0.42	**0.429**	0.444	0.458	0.47
PPMI	0.473	**0.512**	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: summary of all results

4.6.3 Baseline Representations

In Table 4.7 all variations of the baseline representations used directly as input to Decision Trees and SVM's are shown. These examples that do not apply our methodology, serve as a reference point for what is possible using standard linear models without the need for interpretability. In the representations, there is a big performance drop when going from depth three trees to depth one trees. These kind of performance drops are expected for these representations, as they do not have dimensions that correspond to key semantics, so it is unlikely that a smaller tree can use the available dimensions to model a class with limited depth. In this full table the precision and recall scores are included for clarity, mainly to explain why the high recall scores occur. This is because the weights are balanced as a hyper-parameters, and when the weight is balanced so that positive instances are weighted more heavily, the model prioritizes recall over precision. When this high recall score doesn't occur, that means that not balancing the weights performed better on the development data.

The size of the space is not as influential as the representation type in these results for the Decision Trees. For this reason only the best performing representation of each type are shown in Table 4.7. Out of the space-types, PCA performed much better than its counterparts for reuters, newsgroups and sentiment. The MDS representation performs comparably well using a unrestricted depth tree or an SVM, which shows that with a classifier that can make use of all the dimensions, the performance does not decrease as much. This is likely due to the way that PCA orders its dimensions in importance, resulting in key semantics in its first dimensions, giving it an advantage in low-depth Decision Trees. However, this does not necessarily mean that it contains better directions. 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 often encoding the key semantics across more dimensions than other representations, our method is still able find meaningful directions from this space. There is little link between performance on the raw dimensions of the space and performance with rankings on directions in low-depth Decision Trees. This is somewhat counterintuitive, as it would be normal to expect that a representation which performs poorly when used directly as input to a classifier would have similar performance after a linear transformation, but the reason that it works in our case is because low-depth Decision Trees rely on key semantics being disentangled into individual dimensions. Despite the information

encoded in the space, if it is not disentangled then the classifier will not perform well.

Newsgroups	D1				D2				D3				DN				SVM			
	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	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	0.432	0.969	0.612	0.746	0.519
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	0.438	0.969	0.586	0.768	0.474
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	0.438	0.966	0.52	0.745	0.399
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	0.328	0.961	0.468	0.641	0.369
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	0.317	0.865	0.4	0.265	0.812
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	0.316	0.842	0.362	0.233	0.819
MDS 200	0.581	0.184	0.103	**0.837**	0.742	0.262	0.16	0.729	0.719	0.236	0.139	0.785	0.935	0.327	0.332	0.323	0.965	0.501	**0.802**	0.364
MDS 100	0.586	0.187	0.105	0.833	0.754	0.261	0.159	0.727	0.705	0.236	0.138	**0.808**	0.935	0.33	0.338	0.321	0.878	0.439	0.308	0.765
MDS 50	0.593	0.153	0.087	0.647	0.716	0.25	0.15	**0.756**	0.736	0.243	0.144	0.774	0.935	0.324	0.335	0.313	0.854	0.394	0.259	0.821
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	0.346	0.961	0.468	0.641	0.369
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	0.361	**0.971**	**0.628**	0.761	0.535
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	0.376	0.97	0.601	0.758	0.497
PPMI	**0.948**	0.33	**0.532**	0.239	0.947	0.407	0.511	0.338	0.944	**0.444**	0.506	0.396	**0.951**	**0.494**	**0.496**	**0.492**	0.962	0.613	0.627	0.599
Topic	0.852	**0.431**	0.304	0.743	**0.96**	**0.423**	**0.604**	0.326	**0.961**	0.444	**0.606**	0.35	0.944	0.432	0.434	0.429	0.879	0.46	0.318	**0.835**

Table 4.7: Full results for the newsgroups.

Table 4.8: Results for all other domains for the representations.

Reuters	D1		D2		D3		DN		SVM	
	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
AWV	0.782	0.252	0.971	0.328	0.974	0.417	0.973	0.495	0.987	0.719
MDS	0.791	0.263	0.9	0.357	0.979	0.489	0.976	0.522	0.988	0.67
D2V	0.818	0.268	0.867	0.298	0.974	0.445	0.971	0.482	0.986	0.724
PPMI	**0.975**	**0.616**	**0.978**	**0.699**	**0.98**	**0.723**	**0.984**	**0.746**	**0.99**	**0.8**
Topic	0.92	0.411	0.977	0.527	0.977	0.536	0.977	0.56	0.95	0.513

Sentiment	D1		D2		D3		DN		SVM	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
PCA	0.745	0.705	0.755	0.77	0.778	0.773	**0.781**	**0.779**	**0.891**	**0.893**
AWV	0.642	0.652	0.643	0.694	0.695	0.717	0.66	0.663	0.827	0.829
D2V	0.642	0.664	0.66	0.707	0.702	0.7	0.711	0.708	0.878	0.878
PPMI	0.616	0.7	0.655	0.719	0.675	0.73	0.712	0.71	0.887	0.888
Topic	**0.793**	**0.79**	**0.794**	**0.791**	**0.81**	**0.811**	0.733	0.73	0.815	0.822

Placetypes OpenCYC	D1		D2		D3		DN		SVM	
	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
AWV	0.625	**0.383**	0.651	0.376	0.728	**0.396**	**0.844**	**0.362**	0.85	0.466
MDS	0.624	0.364	0.7	**0.397**	0.731	0.374	0.843	0.305	0.861	**0.476**
PPMI	**0.728**	0.371	0.75	0.351	0.739	0.352	0.843	0.323	**0.9**	0.366
Topic	0.708	0.365	**0.87**	0.271	**0.87**	0.313	0.831	0.313	0.808	0.407

Movies Genres	D1		D2		D3		DN		SVM	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
PCA	0.722	0.301	0.755	0.339	0.717	0.321	0.884	0.372	**0.925**	0.518
AWV	0.679	0.29	0.774	0.321	0.756	0.343	0.873	0.312	0.922	0.496
MDS	0.679	0.298	0.79	0.358	0.773	0.354	0.887	0.385	0.875	**0.532**
PPMI	**0.852**	**0.429**	**0.91**	0.443	**0.912**	**0.483**	0.882	**0.416**	0.923	0.526
Topic	0.767	0.415	0.905	**0.472**	0.912	0.455	**0.889**	0.415	0.843	0.491

Placetypes Foursquare	D1		D2		D3		DN		SVM	
	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
AWV	0.767	0.401	0.828	0.478	0.85	0.452	0.905	**0.505**	0.923	**0.622**
MDS	**0.915**	0.438	0.804	0.427	0.86	0.454	0.893	0.462	0.932	0.619
PPMI	0.889	0.473	0.915	**0.512**	0.904	0.491	0.881	0.31	**0.938**	0.567
Topic	0.864	**0.488**	**0.916**	0.433	**0.917**	**0.526**	**0.907**	0.464	0.916	0.569

Movies Keywords	D1		D2		D3		DN		SVM	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
PCA	0.647	0.185	0.644	0.193	0.677	0.199	0.846	0.161	0.787	0.272
AWV	0.5	0.16	0.641	0.179	0.595	0.174	0.853	0.141	0.717	0.23
MDS	0.633	0.179	0.69	0.198	0.674	0.201	0.84	0.163	0.788	**0.28**
PPMI	**0.818**	**0.243**	0.745	**0.224**	0.739	**0.224**	0.847	**0.17**	**0.921**	0.217
Topic	0.629	0.189	**0.932**	0.05	**0.93**	0.075	**0.857**	0.152	0.678	0.21

Placetypes Geonames	D1		D2		D3		DN		SVM	
	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
AWV	0.657	0.326	0.755	0.323	0.842	**0.367**	0.813	0.332	0.865	**0.514**
MDS	0.626	0.349	0.695	**0.34**	0.796	0.272	**0.845**	0.295	0.638	0.397
PPMI	**0.808**	0.361	0.732	0.301	0.76	0.242	0.83	0.283	**0.894**	0.312
Topic	0.771	**0.365**	**0.863**	0.3	**0.85**	0.219	0.828	**0.348**	0.819	0.349

Movies Ratings	D1		D2		D3		DN		SVM	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
PCA	**0.65**	0.463	0.681	**0.475**	0.684	**0.486**	0.744	0.408	0.771	0.58
AWV	0.601	0.423	0.618	0.433	0.596	0.448	0.736	0.372	0.73	0.532
MDS	0.592	0.437	0.635	0.449	0.631	0.452	**0.752**	**0.412**	0.773	**0.589**
PPMI	0.583	0.47	0.635	0.453	0.605	0.453	0.73	0.384	**0.825**	0.536
Topic	0.575	**0.473**	**0.789**	0.243	**0.789**	0.38	0.739	0.375	0.704	0.501

4.6.4 Word Directions

Although Linear SVM's perform the best on these representations without the need for interpretability, other results will be for low-depth Decision Trees in-order to easily distinguish the degree to which key semantics correspond to dimensions in the representations.

The main takeaway from this section is that in most cases performance greatly increases compared to the original representations used directly as input to the model (For the exact differences, see Appendix 7.1).

Interestingly, there was also more variance in the difference between space-type sizes, making it an important hyper-parameter for the single directions. The best space type also varied across domains. Loosely, it is possible to attribute the performance increase for a space-type to either modelling the rankings for the same directions better, or containing unique terms that were particularly relevant to the classes. However, when looking at the qualitative results, generally the words common to all space-types are the most salient 4.3. We can see if this is the case by looking at the Decision Trees for the same task that had the most difference between the space-types and space-sizes. If a Decision Tree contains mostly similar words, but the performance is greater, we can attribute it to a better quality ranking in the space. If the Decision Tree contains different words, especially as the first node, then we know that it was because the words that were modelled well were different between them.

We see that generally, the best space type is the same across a variety of tasks in the same domain, AWV is the best for the place-types but MDS is best for the movies (despite a marginal difference in the ratings). This could mean that performance on one natural task will generalize well to the others, so the space-type/size of the space that we identify contains the key semantics for that domain rather than a particular task.

NDCG was selected as the best score-type for Sentiment, Newsgroups, Reuters, Movies Genres, Movies Keywords in depth-3 Decision Trees. Place-types foursquare used F1-score, but the classes are very unbalanced and there are few documents.

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	**0.963**	0.512	0.694	0.406
PCA 50	0.957	0.373	0.417	0.337	**0.963**	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	**0.792**	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	**0.959**	**0.418**	**0.543**	0.339	0.962	0.465	0.669	0.357	0.962	0.493	**0.707**	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	**0.683**	0.358	0.962	0.494	0.69	0.385
D2V 100	0.844	0.374	0.243	0.803	0.961	**0.49**	0.642	0.396	0.962	0.517	0.67	0.421
D2V 50	0.845	0.388	0.252	**0.844**	0.962	0.488	0.639	0.395	0.963	**0.537**	0.673	**0.446**

Reuters	D1		D2		D3		Sentiment	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	**0.797**	**0.814**	0.802	0.805
AWV	0.975	0.598	0.979	0.656	0.98	0.66	AWV	0.7	0.699	0.711	0.736	0.723	0.735
MDS	0.975	**0.678**	**0.98**	**0.706**	**0.982**	**0.72**	D2V	**0.776**	**0.784**	0.782	0.801	**0.822**	**0.821**
D2V	**0.977**	0.583	0.979	0.664	0.98	0.632							
Placetypes	D1		D2		D3		Movies	D1		D2		D3	
OpenCYC	ACC	F1	ACC	F1	ACC	F1	Genres	ACC	F1	ACC	F1	ACC	F1
PCA	0.632	0.371	0.704	0.381	0.735	0.365	PCA	0.824	0.412	0.82	0.441	0.913	0.463
AWV	**0.66**	**0.404**	**0.734**	**0.428**	**0.755**	**0.39**	AWV	0.81	0.421	0.837	0.436	0.912	0.457
MDS	0.658	0.374	0.711	0.385	0.746	0.35	MDS	**0.849**	**0.446**	**0.839**	**0.463**	**0.918**	**0.495**
Foursquare	ACC	F1	ACC	F1	ACC	F1	Keywords	ACC	F1	ACC	F1	ACC	F1
PCA	0.785	0.477	**0.907**	0.474	0.869	**0.531**	PCA	0.737	0.225	0.727	0.227	**0.709**	0.22
AWV	**0.918**	**0.541**	0.881	**0.498**	0.889	0.466	AWV	0.656	0.201	0.672	0.203	0.652	0.2
MDS	0.82	0.416	0.879	0.482	**0.897**	0.485	MDS	**0.745**	**0.23**	**0.74**	**0.233**	0.708	**0.224**
Geonames	ACC	F1	ACC	F1	ACC	F1	Ratings	ACC	F1	ACC	F1	ACC	F1
PCA	0.665	0.348	0.754	0.342	0.743	0.306	PCA	**0.647**	**0.466**	**0.721**	**0.499**	0.681	0.492
AWV	**0.711**	**0.444**	**0.795**	**0.533**	**0.802**	**0.473**	AWV	0.646	0.463	0.692	0.474	0.677	0.483
MDS	0.591	0.289	0.772	0.333	0.764	0.352	MDS	0.62	0.463	0.692	0.489	**0.686**	**0.498**

Table 4.9: all dirs

4.6.5 Clustered Directions

??

Clustering has two main goals: The first is that features that align with conceptual domain knowledge 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 new more conceptual cluster features perform quantitatively at key domain tasks.

These results were obtained by taking the single directions that performed the best in the previous results and clustering them with a variety of hyper-parameters for the clusters. K-means mostly outperforms Derrac. It does not in the case of Keywords, where it 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.

When looking into the how the individual classes fared, the 100-size Derrac 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 in the best performing k-means 200-size 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's possible that as the Derrac method combined these together into their own cluster they were able to better capture the classes for "shot-in-the-chest" and "machine-guns" because these things occurred in war films where people were shot or shooting. So in this case, the parameters chosen for Derrac supported the classification of the documents into keywords because they better captured particular class concepts through a lesser degree of disentanglement. 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 having a heavily populated cluster average their direction performs better than strongly disentangling the concepts.

Meanwhile, this same lack of disentanglement caused it to lose performance in the "sequel" class. In K-means, the cluster was found for ("franchise sequels sequel installments") while in Derrac the cluster was ("franchise sequels sequel instalments entry returns"). This cluster was also chosen in Derrac 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 Derrac clusters being 100-size (meaning the clusters would contain more terms) and the k-means being 200-size. 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 disentangled with a single cluster corresponding to ("war soldiers military solider army sergeant sgt platoon infantry"). In conclusion, Derrac for the Keywords task captured certain concepts 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 disentangled 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".

In conclusion, the clustering method that performs the best for a task in this unsupervised context is the one that creates clusters that correspond closely with the task's classes, through clustering together words which average into a particular concept, or disentangling words into concepts so that they more precisely model it.

Newsgroups	D1				D2				D3			
	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec	ACC	F1	Prec	Rec
K-means 200	**0.852**	**0.394**	**0.261**	0.795	**0.958**	**0.433**	**0.58**	0.345	**0.963**	**0.513**	**0.704**	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	**0.412**
K-means 50	0.834	0.381	0.248	**0.819**	0.815	0.336	0.212	**0.81**	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		ACC	F1	ACC	F1	ACC	F1
K-means	**0.875**	**0.338**	**0.975**	**0.54**	0.973	**0.58**	K-means	0.623	0.674	**0.837**	**0.844**	0.658	0.707
Derrac	0.797	0.291	0.973	0.402	**0.974**	0.485	Derrac	**0.712**	**0.735**	0.802	0.82	**0.803**	**0.813**
Placetypes	D1		D2		D3		Movies	D1		D2		D3	
OpenCYC	ACC	F1	ACC	F1	ACC	F1	Genres	ACC	F1	ACC	F1	ACC	F1
K-means	**0.641**	**0.413**	**0.735**	**0.405**	0.75	**0.43**	K-means	**0.813**	**0.431**	**0.913**	**0.513**	**0.913**	**0.506**
Derrac	0.605	0.39	0.672	0.392	**0.755**	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	**0.913**	**0.462**	**0.911**	**0.5**	**0.891**	**0.511**	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	**0.726**	**0.215**	**0.745**	**0.22**	**0.707**	**0.219**
Geonames	ACC	F1	ACC	F1	ACC	F1	Ratings	ACC	F1	ACC	F1	ACC	F1
K-means	**0.772**	0.43	**0.774**	0.407	**0.819**	**0.472**	K-means	**0.671**	**0.504**	0.638	**0.507**	**0.686**	**0.513**
Derrac	0.678	**0.449**	0.74	**0.411**	0.807	0.415	Derrac	0.651	0.445	**0.669**	0.463	0.627	0.479

Table 4.11: The best clustering results for each domain and task

4.6.6 Conclusion

In conclusion, we introduce a methodology to go from a Vector Space Model of Semantics and an associated bag-of-words to an interpretable representation and interpretable classifiers. We define an interpretable representation in this work as having two properties: disentanglement and labels, and an interpretable classifier as a simple linear classifier that has components corresponding to the interpretable representation that has these properties, e.g. nodes in a decision tree. In general, we give a simple methodology that can be used to achieve interpretable features and classifiers as an alternative to methods like Topic Models, and give insight into the parameters required and qualitative results that can be obtained. We extensively test the qualitative and quantitative results, finding that the highest-performing quantitative results also make good intuitive qualitative sense. We find that our method greatly outperforms the original representations on low-depth Decision Trees, giving good evidence that we have disentangled the representation. Additionally, we find that we are also competitive with standard interpretable representation baselines in most cases. We introduce variations to the original work that produced these kind of interpretable representations, in particular finding that scoring directions using NDCG performed better than Kappa in most cases, and that we could achieve much stronger results than the original clustering method using K-means. Further, we experimented using a variety of space-types and domains, verifying that the methodology can be applied more generally than shown in [6]. 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 our cluster labels fare in real-world situations.

Chapter 5

Fine-tuning Vector Spaces to Improve Their Directions

5.1 Introduction

Chapter 4 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 6.1 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 5.1, a problem that can occur with feature-directions from representations learned with a similarity-centred objective, e.g. Multi-Dimensional Scaling (see Section **??**) 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

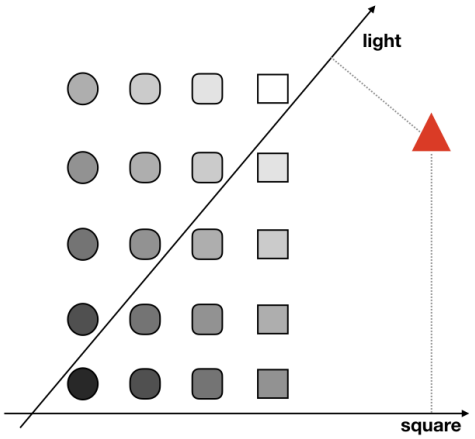

Figure 5.1: Toy example showing the effect of outliers in a two-dimensional embedding of geometric shapes..

“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 5.1: 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 5.1, 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.

5.2 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 $\preccurlyeq_f$ such that $o \preccurlyeq_f o'$ iff the feature f is more prominent in the BoW represention of object o' than in the BoW representation of o. We will refer to $\preccurlyeq_f$ 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 $o \preccurlyeq o'$ 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.

5.2.1 Generating Target Rankings

Let $C_1, ..., C_K$ be the clusters that were found using the method from Section 4.4.1. Each cluster C_i typically corresponds to a set of semantically related words $\{w_1, ..., w_n\}$, 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 $C = \{w_1, ..., w_m\}$, 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 $\mathit{ppmi}(C, o)$ for the PPMI score corresponding to this BoC representation, which is evaluated in the same way as $\mathit{ppmi}(C, o)$, 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 $\mathit{ppmi}(C_i, o_1) > \mathit{ppmi}(C_i, o_2)$. 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.

5.2.2 Generating Target Feature Values

Finding directions in a vector space that induce a set of given target rankings is computationally hard[1]. Therefore, rather than directly using the target rankings from Section 5.2.1 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 $\mathit{ppmi}(C_j, o_i)$. 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

[1]It is complete for the complexity class $\exists\mathbb{R}$, which sits between NP and PSPACE [31].

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 $o_{\sigma_1}, ..., o_{\sigma_n}$ be an enumeration of the objects such that $\mathit{ppmi}(C_j, o_{\sigma_i}) \leq \mathit{ppmi}(C_j, o_{\sigma_{i+1}})$ for $i \in \{1, ..., n-1\}$. The corresponding target values $\tau(C_j, o_i)$ are then obtained by solving the following optimization problem:

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

Subject to:

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

5.2.3 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 $o_1, ..., o_n \in \mathbb{R}^k$. 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 ith 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 $g_i = (g_i^1, ..., g_i^K)$ 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 5.1, 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	**0.50**	**0.47**	**0.44**
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 AWVw	0.41	0.41	0.43
AWVw	0.38	0.40	0.43
LDA	0.40	0.37	0.35

Table 5.2: Results for 20 Newsgroups.

5.3 Quantitative Evaluation

To evaluate our method, as in Chapter 5 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	**0.57**	**0.56**	0.51	FT MDS	**0.33**	**0.33**	0.24	FT MDS	**0.49**	**0.51**	**0.46**
MDS	0.40	0.49	**0.52**	MDS	0.31	0.32	**0.25**	MDS	0.46	0.49	**0.46**
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	**0.37**	0.28
AWV	0.28	0.28	0.22	AWV	0.32	0.37	0.31	AWV	0.33	0.35	0.26
LDA	**0.34**	**0.32**	**0.27**	LDA	**0.55**	**0.48**	**0.47**	LDA	**0.40**	0.36	**0.31**

Table 5.3: The results for Movie Reviews and Place-Types on depth-1, depth-3 and unbounded trees. .

In Chapter 5 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	**0.82**	**0.80**
FT AWV	0.72	0.76	0.71
AWV	0.74	0.76	0.71
LDA	**0.79**	0.80	0.79

Table 5.4: 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 5 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 datasedt, 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 [7],

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

For the cluster size, we follow work by Steven Schockaert[31] 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.

5.3.1 Results

Table 5.2 shows the results for the 20 newsgroups dataset, where we use FT to indicate the results with fine-tuning[2]. 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 AWV 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 5.3. For the MDS representations, the fine-tuning method again consistently improved the results for D1 and D3 trees. For the AWV 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 $\{gore, gory, horror, gruesome\}$.

[2]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 5.4. 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.

5.4 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.

Chapter 6

Directions in Neural Networks

6.1 Introduction

6.2 Background

There are two typical methods for making neural networks interpretable, explaining the neural network after it has been learned and modifying the way the neural network is learned such that it is interpretable. However, our method does not easily fit into either of these categories, as we do not explain the neural network in terms of its predictions or making it so that it learns an additional interpretable objective.

Interpretability and completeness

Interpretability of information inside the network versus predictions

Post-hoc versus baking it into the model

Creating networks that are easier to explain: Disentangled representations, GAN, PCA< Independent Component Analytsis, Nonnegative matrix factorization, Variational autoencoding, Beta-VAE, InfoGAN, Construction of graphs, decision trees

Deep networks that are trained to make their own explanations

Linear proxy models

Decision trees from neural networks (super deep and bad)

Rule extraction from neural networks

Concept Activation Vectors "are a framework for interpretation of a neural nets representations by identifying and probing driections that align with human interpretable concepts"

Persuasive versus transparent

6.2.1 Explanations

6.2.2 Rules and Neural Networks

6.2.3 Investigation methods

6.2.4 Feed-forward networks

6.2.5 Auto-encoders

6.3 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 be 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 **??** where example clustered features from the best performing decision trees are shown.

6.3.1 Auto-encoders

6.3.2 Feed-forward networks

These 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 changes: Cross-entropy loss, Adagrad trainer with default parameters, Dropout, Softmax activation on the output layer, ReLu or tanH activation function.

We trained neural networks with non-linear activation functions so that they would perform better than linear SVM's on document representations. The assumption behind this is that if the network is able to achieve stronger results on the task than a baseline representation, it must be doing something more than representing the information. In other words, the representation must be better for the task than an unsupervised representation. The goal of this section is to discover why neural networks are able to perform so well, and explain it in terms of directions.

To learn a baseline feedforward network, we take the highest performing representation found in Table **??** for clusters on depth-3 decision trees and use them as input representations. 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 a non-linear 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. These feedforward networks are simple, but do achieve stronger results than the linear SVM.

The differences between the supervised representation and the unsupervised representation are examined here, with the main question being how exactly does a neural network achieve such strong results on the task? Some general ideas or explanations that are given are that through non-linear transformations, the internal representation of the neural network is able to find a more complex or accurate representation of domain knowledge relevant to the task.

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 this neural network performs much more strongly than the one starting from the Doc2Vec space.

6.4 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.

We investigate the task using the newsgroups as it gives a good example of a representation that contains many different concepts and aspects of domain knowledge, as well as a well defined classification task that if achieved well on clearly demonstrates the representation contains a good amount of information in the domain. The place-types task does not contain too many entities, making it unreliable for neural network training and the reuters task has similar problems.

One alternative would be the movies domain, but as identifying the genres of movies is functionally similar to the newsgroups task, we assume that any results found for the newsgroups can easily generalize to any classification task.

We tune the network for the following values:

epoch = [100, 200, 300] activation$_function = [$"$relu$","$tanh$"$]dropout = [0.1, 0.25, 0.5, 0.75]hidden_layer_siz$
$[1, 2, 3, 4]$

When using the bag-of-words as input to the neural network for the newsgroups, we found that it worked well with default parameters. The parameters for this network are not tuned as the baseline network is, instead an initial hidden layer of size 1000 was chosen following recommendations in [**?**] and the secondary layer was chosen as it has a size similar to the Doc2Vec space and converged quickly. As the method described in chapter 5 scales poorly to larger dimensional spaces, this secondary layer was necessary. 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.

6.4.1 Feed-forward Networks

6.5 Quantitative Results

Here, we show quantitative results for:

- 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 can perform well on the task or sometimes better. This validates the idea that interpretable features obtained from the neural network using the method in Chapter 5 can be used as an

alternative in a myriad of simple interpretable classifiers without much loss in prediction power. As these features are able to perform well on a simple classifier, it additionally validates the idea that these interpretable features can be used to investigate the neural network and the domain, which is looked at in more depth in the Qualitative Results Section **??**.

This section additionally provides insight into the neural networks using the quantitative results. If the neural network performs better than a linear svm with the unsupervised representation as input, it gives some insight into the neural network, in particular that it must be arranging the space in a better way than the unsupervised representation. Similarly if it performs poorly, it must be arranging the space poorly. These results are preliminary insights that inform what we expect the directions to be like in in Section **??**.

Neural Network Results

For the newsgroups, the neural network with bow input performed significantly better than the one with a vector space as input.

Compared to newsgroups, the neural network that used a vector space as input performed significantly better than the linear SVM on the unsupervised representation. It obtained an F1-score of 0.574, when the linear SVM's score was 0.532. The assumption follows that if it actually has found more meaningful concepts in the space, then the directions should also perform well on a depth-3 decision tree.

For the place-types domain, the neural network that used a vector space as input performed significantly worse than the SVM, as it scored 0.53 in F1 score compared to 0.622. However, this is not particularly meaningful as there are so few entities.

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.

content...
Entities, terms that have gone down the most from the original space, terms that have gone up the most

Directions

Clusters

6.6 Qualitative Investigation

6.6.1 Feed-forward networks

Neural networks with vector space as input

Generally, we see a similar trend in other domains. If noise is in the original space, using it as input to a neural network will not remove that noise. In that case, what changes between the original vector space and the neural network? To answer this question, the terms arranged by the difference between them is listed in table

Neural networks with bow as input

However, this only covers those terms which were also in the original space. There are also unique word directions identified in the bow representation.

To demonstrate this, we show a comparison of the neural network with bow input representation and the original representation each projected using PCA into two dimensions.

The difference between the rankings of common single word directions is also investigated. Hypothetically, the rankings should make more intuitive sense, and noise should be removed, if the neural network does indeed prioritize the rankings of this important domain knowledge.

In the movie review domain, the original MDS space had high NDCG scores for noise, e.g. "berardinelli" a reviewer's name, or "listinfo" which is review metadata. This was also present in the neural network representation where this vector space was used as input. However, it was

content...

content...
Horror: Most important clusters used for space 1, most important for space 2

not present when starting from a bag-of-words. This is interesting, because the neural network performed worse with the bag-of-words as input, and very well with the vector space as input, despite their directions performing about the same on a depth-3 decision tree.

Decision trees of depth-3 in the quantiative results performed as well as neural networks, however the trees obtained from this representation are simplistic in the form of "if HORROR then HORROR" rather than representing some complex domain knowledge.

There are common word directions, e.g. horror. But how have the rankings changed for these directions? This is investigated by looking at the biggest differences in rankings for the same word between the original representation and the fine-tuned representation. We use the place-types results for this as they have easy to understand entities. We can see that e.g.

The way that knowledge is represented is changed significantly in the neural network with bow input compared to the original representation. This can be verified by what clusters are used in the decision trees to classify the entities. For the neural network with bow, But what if these directions are not new concepts, but instead the same rankings but with different words labelling them?

Directions

Two groups of single word directions and clusters were obtained from three different domains, in each one is from the second layer of a two-layer neural network starting from a bag-of-words and the other is from the first layer of a one-layer neural network starting from an unsupervised representation. The examples in this section for the clusters are from the clusters that performed the best in depth-3 decision-trees. When investigating single word directions, sometimes it is unclear what they mean. In this case, the most similar single word-directions are found and used to provide additional context.

Entities, terms that have gone down the most from the original space, terms that have gone up the most

Starting from vector space

Starting from bag-of-words

Obtaining directions from this space takes much longer than normal vector spaces.

6.6.2 Auto-encoders

6.7 Conclusion

Neural network models that encode spatial relationships in their hidden layers have achieved state-of-the-art in Text Classification by using transfer learning from a pre-trained Language Model [9]. There have also been neural network models that produce an interpretable representation, for example InfoGan. Most state-of-the-art results rely on Vector Space Models. Ideally the method would be able to achieve strong results for simple interpretable classifiers by transforming an existing representation that performs well at the task.

6.7.1 Chapter 3 Space Types

	Genres			Keywords			Ratings		
Movies	D1	D2	D3	D1	D2	D3	D1	D2	D3
Space	50 PCA	50 MDS	100 MDS	200 PCA	200 MDS	200 MDS	50 PCA	200 PCA	50 PCA
Single directions	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Newsgroups			Sentiment			Reuters		
Rep	200 PCA	200 PCA	100 PCA	PCA 100	PCA 50	PCA 50	200 PCA	200 PCA	100 PCA
Single dir	200 MDS	100 D2V	50 D2V	D2V 100	PCA 50	D2V 100	N/A	N/A	N/A
	Foursquare			OpenCYC			Geonames		
Placetypes	D1	D2	D3	D1	D2	D3	D1	D2	D3
Rep	MDS 100	AWV 50	MDS 200	AWV 50	MDS 200	AWV 50	MDS 50	MDS 50	AWV 200
Single dir	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 6.1: Space-types, clusters have the same as single directions.

Theoretically if we are able to identify features from each layer, then we can also map how they interact with each other.

There are no additional techniques applied to improve the performance of the neural networks.

Chapter 7

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	**0.149**
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	**0.109**	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	**0.378**	**0.234**	**0.439**	-0.498	**0.22**	0.203	0.509	-0.372	0.243	**0.257**	**0.568**	-0.406
MDS 100	0.271	0.178	0.138	-0.108	0.205	0.167	0.465	-0.401	**0.254**	0.217	0.506	-0.459
MDS 50	0.228	0.171	0.119	**0.115**	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	**0.514**	-0.288	0.192	0.225	0.526	-0.365
D2V 100	0.162	0.166	0.123	0.041	0.169	**0.222**	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

Reuters	D1		D2		D3		Sentiment	D1		D2		D3	
	ACC	F1	ACC	F1	ACC	F1		ACC	F1	ACC	F1	ACC	F1
PCA	0.129	0.33	0.062	0.265	-0.002	-0.034	PCA	-0.006	0.053	0.042	0.044	0.024	0.032
AWV	**0.193**	0.345	0.008	0.327	**0.007**	**0.243**	AWV	0.057	0.047	0.068	0.042	0.028	0.018
MDS	0.184	**0.414**	0.08	0.349	0.003	0.231	D2V	**0.134**	**0.12**	**0.122**	**0.094**	**0.12**	**0.121**
D2V	0.159	0.316	**0.112**	**0.366**	0.006	0.188							
Placetypes	D1		D2		D3		Movies	D1		D2		D3	
OpenCYC	ACC	F1	ACC	F1	ACC	F1	Genres	ACC	F1	ACC	F1	ACC	F1
PCA	**0.047**	**0.025**	-0.003	0.038	**0.04**	**0.024**	PCA	0.102	0.111	**0.064**	0.101	**0.196**	**0.142**
AWV	0.036	0.021	**0.083**	**0.052**	0.027	-0.006	AWV	0.132	0.132	0.064	**0.115**	0.156	0.114
MDS	0.034	0.009	0.011	-0.012	0.016	-0.024	MDS	**0.17**	**0.148**	0.049	0.104	0.145	0.141
Foursquare	ACC	F1	ACC	F1	ACC	F1	Keywords	ACC	F1	ACC	F1	ACC	F1
PCA	0.054	0.135	**0.084**	**0.082**	0.008	**0.143**	PCA	0.09	0.04	**0.083**	0.034	0.032	0.022
AWV	**0.151**	**0.14**	0.053	0.02	**0.038**	0.014	AWV	**0.156**	0.041	0.031	0.024	**0.057**	**0.025**
MDS	-0.094	-0.022	0.075	0.055	0.038	0.031	MDS	0.111	**0.051**	0.05	**0.035**	0.033	0.023
Geonames	ACC	F1	ACC	F1	ACC	F1	Ratings	ACC	F1	ACC	F1	ACC	F1
PCA	**0.163**	0.047	0.063	0.037	**0.063**	0.011	PCA	-0.003	0.003	0.04	0.023	-0.003	0.007
AWV	0.054	**0.119**	0.04	**0.21**	-0.039	**0.106**	AWV	**0.045**	**0.041**	**0.074**	**0.042**	**0.08**	0.036
MDS	-0.035	-0.06	**0.078**	-0.007	-0.032	0.08	MDS	0.028	0.026	0.057	0.04	0.055	**0.045**

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)		Keywords (3)		Keywords (4)		Keywords (5)		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	drunkenness	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
corpse	1008	fistfight	977	machine-gun	878	sequel	801			bop	104
crying	1149	flashback	1937	male-nudity	1122	sex	2126			gnp	136
cult-film	1636	friend	1193	marriage	1407	shootout	1174			dlr	162
dancer	1020	friendship	1903	martial-arts	824	shot-in-the-chest	892			money-supply	168

Table 7.2: Positive Instance Counts for each Class

GNU Free Documentation License

Version 1.2, November 2002

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. Secondarily, 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.

Bibliography

[1] David Arthur and Sergei Vassilvitskii. k-means ++ : The Advantages of Careful Seeding. 8:1–11.

[2] Andrew M. Dai, Christopher Olah, and Quoc V. Le. Document embedding with paragraph vectors. *CoRR*, abs/1507.07998, 2015.

[3] Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian LDA for topic models with word embeddings. In *Proc. ACL*, pages 795–804, 2015.

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

[5] J. Derrac and S. Schockaert. Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning. *Artificial Intelligence*, pages 74–105, 2015.

[6] Joaqu??n Derrac and Steven Schockaert. Inducing semantic relations from conceptual spaces: A data-driven approach to plausible reasoning. *Artificial Intelligence*, 228:66–94, 2015.

[7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[8] Peter Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT press, 2004.

[9] Chengyue Gong. FRAGE : Frequency-Agnostic Word Representation. 1(Nips):1–15, 2018.

[10] Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian PadÃ^3. $Distributionalvectorsencodereferentialattributes. In Proceedingsofthe 2015 Conferenceon$

[11] Ulrike Hahn and Nick Chater. Similarity and rules: distinct? exhaustive? empirically distinguishable? *Cognition*, 65:197 – 230, 1998.

[12] Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, 2016.

[13] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[14] Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. MEmbER : Max-Margin Based Embeddings for Entity Retrieval.

[15] Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. Member: Max-margin based embeddings for entity retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 783–792, 2017.

[16] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[17] Joo-kyung Kim. Deriving adjectival scales from continuous space word representations. (October):1625–1630, 2013.

[18] Adriana Kovashka, Devi Parikh, and Kristen Grauman. WhittleSearch : Image Search with Relative Attribute Feedback.

[19] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Image search with relative attribute feedback. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2973–2980, 2012.

[20] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1188–1196, 2014.

[21] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Proc. AAAI*, pages 2418–2424, 2015.

[22] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. pages 142–150, 2011.

[23] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. Explain Images with Multimodal Recurrent Neural Networks. *arXiv:1410.1090 [cs]*, pages 1–9, 2014.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th*

International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[25] Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. How do Humans Understand Explanations from Machine Learning Systems? An Evaluation of the Human-Interpretability of Explanation. pages 1–21, 2018.

[26] Hamid Palangi, Paul Smolensky, Xiaodong He, and Li Deng. Question-Answering with Grammatically-Interpretable Representations. 2017.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014.

[28] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[29] Sascha Rothe and Language Processing. Word Embedding Calculus in Meaningful Ultradense Subspaces. pages 512–517, 2016.

[30] T.L. Saaty and M.S. Ozdemir. Why the magic number seven plus or minus two. *Mathematical and Computer Modelling*, 38(3):233–244, 2003.

[31] Steven Schockaert and Jae Hee Lee. Qualitative reasoning about directions in semantic spaces. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3207–3213. AAAI Press, 2015.

[32] D Sculley. Web-Scale K-Means Clustering. pages 4–5, 2010.

[33] Amos Tversky. Features of similarity. *Psychological review*, 84:327–352, 1977.

[34] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 165–174, 2016.

[35] Paolo Viappiani. Preference-based Search using Example-Critiquing with Suggestions. 27:465–503, 2006.

[36] Paolo Viappiani, Boi Faltings, and Pearl Pu. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research*, 27:465–503, 2006.

[37] Jesse Vig, Shilad Sen, and John Riedl. The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems*, 2(3):13:1–13:44, 2012.

[38] Jesse Vig, Shilad Sen, and John Riedl. The Tag Genome : Encoding Community Knowledge to Support Novel Interaction The Tag Genome : Encoding Community Knowledge to Support. (November), 2014.

[39] Youwei Zhang and Laurent El Ghaoui. Large-Scale Sparse Principal Component Analysis with Application to Text Data. pages 1–8, 2012.