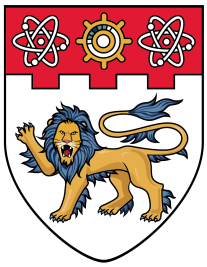

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

# Interpretable Vector Language Models


**ENG JING KEAT**
U1940310B

Supervisor: Dr. Fedor Duzhin


A Final Year Report submitted to Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University in partial fulfilment of the honours requirements for the Degree of

Bachelor of Science in Mathematical Sciences

Academic Year 2022 Semester 2

# Abstract

Natural Language Processing (NLP) is a branch of computer science that focuses on the development of algorithms for understanding, interpreting, and generating human language texts. A crucial technique in NLP is word embedding, where models such as Word2Vec and GloVe assign vectors to words in a vocabulary such that the Euclidean space structure (norms and angles of word vectors) aligns with the semantic structure of the training corpus. Despite their effectiveness, the individual entries of word embedding models are difficult to interpret due to the simultaneous rotation of all pre-trained word vectors preserves norms and angles while mixing up individual entries. In this study, we proposed a novel approach for generating word embeddings with interpretable entries. To achieve it, we introduced a metric to quantify the interpretability of a word embedding model. Additionally, we connected the interpretability of a word embedding model to a specific loss function defined on the Lie group $SO(d)$. We then compared three loss functions, namely, the Varimax loss function inspired by factor analysis, the $l_1$-norm, and a combination of the two. Our results showed that the Varimax loss function yielded word embeddings with the highest interpretability among the three methods, as it maximizes the sum of the variances of squared entries, enabling successful interpretation of some columns in the resulting word embedding matrices. This study offers insights into generating interpretable word embeddings while preserving semantic structure.

# Acknowledgement

First of all, I would like to express my gratitude to Nanyang Technological Univerisity (NTU), Division of Mathematical Sciences, for providing me the opportunity to complete my Final Year Project (FYP) as part of my undergraduate studies. I am deeply thankful to my supervisor, Dr. Fedor Duzhin, for allowing me to work on this project, and for his valuable guidance and patience, which enabled me to successfully finish the project.

Besides, I wish to extend my appreciation to my friends for their continuous support and encouragement throughout this journey. Finally, I would like to convey my utmost gratitude to my family members for their unwavering love and support, making my pursuit of a Bachelor of Science in Mathematical Sciences in NTU both enjoyable and fulfilling.

# Table of Contents

# Lists of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

Natural Language Processing (NLP) is subfield of computer science that focuses on creating algorithms to understand, interpret, and generate human language texts. NLP has a wide range of applications, such as text categorization, information extraction, and automated language translation [1, 2].

A crucial technique in NLP is word embedding, which uses algorithms like Word2Vec and GloVe to assign vectors to words based on the analysis of large text corpora. These algorithms ensure that the vector space structure aligns with the semantic structure of the language [3, 4]. For instance, the vector representing the word "king" minus the vector representing "man" plus the vector representing "woman" is closest to the vector representing the word "queen" in the embedding space [3].

However, word embeddings have limited interpretability, as individual entries of word vectors do not carry specific meanings. This is because a simultaneous rotation of all vectors still preserves the semantic structure of the language.

Recent studies have attempted to create interpretable word embeddings by incorporating explicit knowledge, such as semantic lexicons or supervised categories, into the embedding process [5, 6]. While these approaches claimed that there was a significant improvement in the interpretability of word embeddings, they do not fully address the issue of generating word embeddings with interpretable entries. Additionally, the number of lexical information types that can be created is limited, as only up to three orthogonal ultradense subspaces can be formed. Hence, this limitation may prevent the approach from capturing all relevant information present in the embedding space.

## 1.2 Objectives and Scope

Hence, is it possible to develop a novel word embedding approach that achieves high performance in NLP tasks and offers improved interpretability compared to existing models? In this study, we will propose a new method for generating word embeddings that yield interpretable vector entries by leveraging the properties of special orthogonal group $SO(d)$, which is a Lie group, and using objective functions associated with rotation techniques used in Factor Analysis [7, 8].

The revised new method will be applied to word embeddings generated from 'The Adventures of Sherlock Holmes' using the GloVe word embedding technique. Interpretable entries will be created by performing orthogonal rotations with three objective functions. Additionally, we design a metric to assess the interpretability of the resulting word embedding matrix. This innovative approach has the potential to improve the usability and interpretability of word embeddings, making them more accessible to researchers and practitioners in industry and academia.

## 1.3 Organization

This report is structured into five chapters. Chapter 2∼4 provides relevant background information and theory to aid in the understanding of the methodology of this study. Chapter 5 outlines the implementation of the techniques and algorithms used in this research. The corresponding results of this study are shown and discussed in Chapter 6. Lastly, the thesis is concluded in Chapter 7 with some concluding remarks and a discussion of prospective future study.

# Chapter 2

# Interpretability

## 2.1 Interpretable Models

Interpretability is defined as *the ability to explain or convey something in a way that a human can understand* [9]. Interpretability is essential in various fields, including Machine Learning (ML), Artificial Intelligence (AI) and data science. The growing importance of interpretability can be attributed to the increasing complexity of recent models, along with the requirement of high reliability, ethical and legal, as well as accountability in decision-making processes [10].

Interpretability is a critical aspect of machine learning, particularly when it comes to complex models such as deep neural networks. These models are often used in applications where high accuracy is essential. However, it can be difficult to understand how the underlying factors driving the model's predictions. Therefore, interpretability is often viewed as a trade-off between model accuracy and explainability [11].

On the other hand, simpler models like decision trees are more interpretable, however, they may not be able to achieve the same levels of accuracy as complex models. Thus, decision trees are widely used in applications when interpretability is important, such as medical diagnosis, where the ability to explain a prediction is vital [12]. Doctors and other healthcare professionals can make better decisions by understanding how a model generates its predictions with the aid of interpretability. Interpretability is therefore a key consideration when choosing the appropriate model for a certain task.

There are several approaches to interpret a model, depending on the type of the model and the particular requirements. One of the common approaches is to use visualization techniques to represent the behavior and outputs of the model. For instance, decision trees can be represented graphically as a tree-like structure, with each node denoting a decision based on a particular feature or attribute of the data. Similarly, deep neural networks can be visualized as a collection of linked nodes and

edges, where each node stands for a neuron and each edge for a connection between neurons.

Another approaches to do interpretation is to use feature importance methods, which aim to identify the features or attributes of the data that are most relevant to the model's predictions. For example, in linear regression models, feature importance can be measured by the magnitude of the coefficients assigned to each feature. In decision trees, feature importance can be measured by the number of times a feature is used to make a split.

In the following sections, various interpretable models will be discussed. An overview of each model will be provided, as well as explanations of how it can be interpreted.

### 2.1.1 Linear Regression Models

A linear regression model is a tool that can predict a outcome value by using a weighted sum of input features. Linear regression models have been used for a long time by various professionals such as statisticians and computer scientists who work with quantitative problems [13].

Linear models are suitable for representing how the target value $y$ is related to certain features $x$. The relationships learned are linear, and can be expressed as an equation as follows for a single instance $i$:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon$$

This approach assumes that the predicted value of $y$ for a given instance $i$ can be expressed as a linear combination of its $p$ features, with each feature multiplied by a learned weight or coefficient $\beta_k$. The intercept $\beta_0$ is a constant term that is not associated with any feature. The difference between the predicted outcome and the exact value is represented by the error term $\epsilon$, which is assumed to be normally distributed, indicating that the errors can be positive or negative and vary in magnitude.

The results of a linear regression model can be interpreted by look at the estimated values of $\beta_0$ and $\beta_k$ for $k = 1, 2, ..., p$. The intercept $\beta_0$ represents the predicted value of $y$ when $x_k$ is equal to 0. The coefficients $\beta_k$ represent the change in $y$ for a one-unit increase in $x_k$.

R-squared measurement is one of the methods to interpret linear models, it is a statistical measure that indicates the proportion of the total variance of the outcome variable that is explained by the linear model. The formula for calculating R-squared involves comparing the amount of unexplained variation (SSE) to the total variation (SST):

$$R^2 = 1 - SSE/SST$$

$$SSE = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

$$SST = \sum_{i=1}^{n} (y^{(i)} - \bar{y})^2$$

A higher R-squared value indicates that the model fits the data better. However, relying solely on R-squared can be misleading since it increases with the addition of more features, even irrelevant ones. Thus, it is more reliable to use adjusted R-squared, which takes into account the number of features in the model:

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

A low adjusted R-squared value implies that the model is not a good fit for the data and does not explain much of the variance. It is essential to note that a negative R-squared value is possible when the model fits the data worse than the mean value of the target variable.

Besides, t-statistic of a feature can be used to evaluate the significance of this feature in a linear regression model. This is calculated by dividing the estimated weight by its standard error. The higher the absolute value of the t-statistic, the more important the feature is used to predict the target variable. On the other hand, if the estimated weight has a large variance, the feature is less important. Therefore, the t-statistic provides a useful tool for determining which features are most influential in the model.

### 2.1.2 Decision Tree

Tree-based methods involve dividing the feature space into a set of rectangles, and then fitting a simple model, such as a constant value, in each of these sections [14]. In large databases, decision trees can be employed to identify significant features and extract patterns that are valuable for predictive modeling and discrimination [15]. These methods are particularly useful when there is a nonlinear relationship between the features and the outcome or when the features interact [13]. One popular type of the tree-based techniques is decision tree, which can be grown using a variety of algorithms that differ in their structure, such as criteria for splitting, stopping rules, and methods for estimating the simple models in the leaf nodes. The classification and regression trees (CART) algorithm is the most commonly used to induce trees, and it will be discussed in the following paragraphs, while the interpretation of other types of trees is similar.

The formula below shows the association between the target variable $y$ and predictor variables $x$:

$$\hat{y} = \hat{f}(x) = \sum_{i=1}^{n} (c_i I\{x \in R_i\})$$

$I\{x \in R_i\}$ is an identity function, it is equal to 1 if $x \in R_i$ and 0 otherwise. Each event belongs to exactly one terminal node. If $x \in R_k$, $\hat{y} = c_k$ where $c_k$ is the mean value of all training observations belongs to $R_k$.

The interpretation of decision trees is straightforward: beginning from the root node, each subsequent node and the edges connecting them indicate the subsets being examined. The predicted outcome is displayed when a leaf node is reached. The edges connecting the nodes are linked by the logical operator 'AND'.

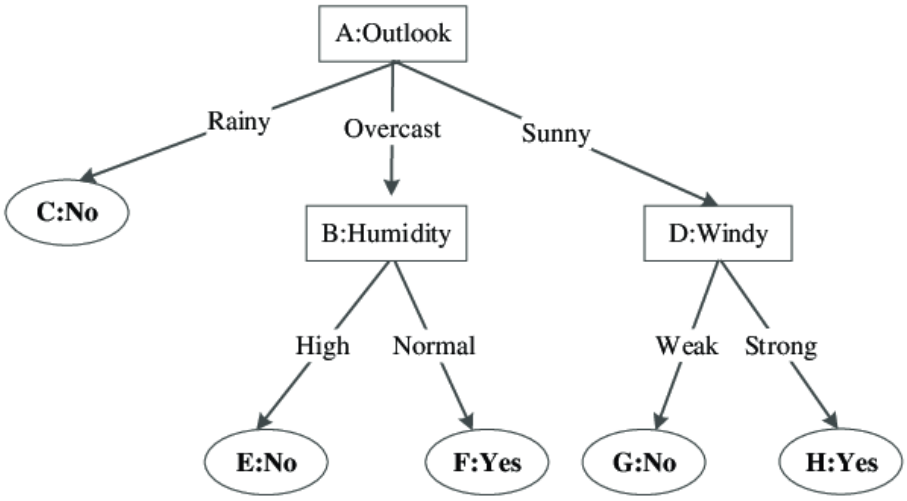


Figure 2-1: A simple decision tree model
Adapted from [16]

As an illustration for figure 2-1, if the weather condition is both sunny and windy, the predicted outcome would be "yes."

## 2.2 Word Embeddings Methods

Word embedding is a technique used in natural language processing to represent words as vectors in a high-dimensional space. In mathematical terms, this is achieved through a function denoted as $f$, which maps a set of $V$ words into word embedding vectors of dimension $d$:

$$f : V \mapsto \mathbb{R}^d$$

If there is a vocabulary of $V$ words and each word is represented by a $d$-dimensional vector, then the word embedding matrix $W$ can be defined as:

$$W = \left[w_1, w_2, ..., w_N\right]^T$$

where each $w_i$ is a $d$-dimensional vector representing the $i$-th word in the vocabulary. We can access the embedding vector for a given word $x$ by looking up the corresponding $x$-th row in the matrix $W$.

The elements of the embedding vector can be learned through various methods, such as optimizing a neural network objective function or applying dimensionality reduction techniques to a co-occurrence matrix of words in a corpus [3, 4].

The significance of model interpretability and methods for interpreting models are explored in the previous section. When it comes to word embeddings, how it can be interpreted? Below are various types of word embeddings techniques, although the interpretability of the entries within these word embeddings remains a challenge, they succeed in effectively demonstrating the semantic relationships between words. For example, these word embeddings can be used to cluster words with similar meanings or to identify analogies between words.

### 2.2.1 Word2Vec

Word2Vec was created by Tomas Mikolov et al. [3] in 2013, which captures contextual and semantic similarity in high-quality, distributed, and continuous dense vector representations of words. It is a collection of associated models to generate word embeddings. Essentially, these models can process huge text corpora and construct a vocabulary of possible words.

The embedding for each word is produced in the vector space which represents that vocabulary. The size of the word embedding vectors may be set, and the total number of vectors is necessarily equal to the vocabulary size. Hence, this vector space has a lower dimension than the vector space built by one-hot vectors. It is expected that similar words are not only close to each other in the vector space, but also show similarity in multiple degrees. The similarity between two word vectors is computed using the cosine similarity measure. The cosine similarity between word $A$ and word $B$ can be calculated using the dot product of their corresponding word embedding vectors divided by the product of their magnitudes, as follows:

$$similarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{d} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{d} A_i^2} \cdot \sqrt{\sum_{i=1}^{d} B_i^2}}$$

We can interpret word embedding which is generated by Word2Vec by performing arithmetic operations on word vectors to observe relationships and analogies between words. For example, $king - man + woman = queen$. Besides, we can measure the similarity between two words by calculating the cosine similarity between their corresponding word vectors. A higher cosine similarity value indicates a stronger semantic relationship between the words.

There are two different model architectures which can be used by Word2Vec to generate a distributed representation of words, which are continuous bag-of-words (CBOW) and continuous skip-gram.

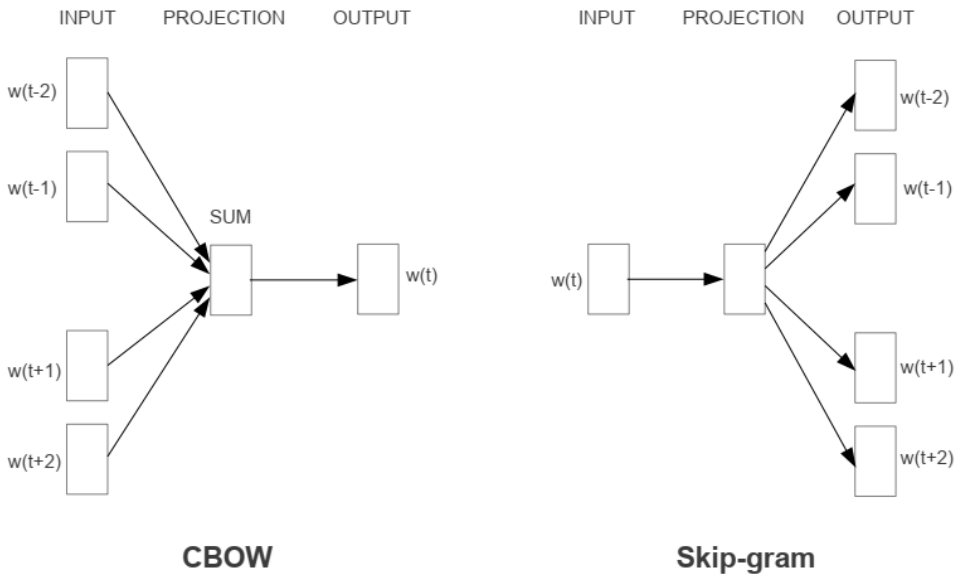


Figure 2-2: Model architectures of Continuous Bag-of-Words model and Skip-gram model

Adapted from [3]

### Continuous Bag-of-Words

The CBOW model architecture takes a fixed-size window of context words around the target word and uses these words to predict the target word, and it was assumed that the prediction will not be influenced by the order of context words.

At the beginning, the number of input context words is *2 * window size*, and then they will be passed to an embedding layer of dimension *vocabulary size * embedding size* to generate word embeddings for each context word. After that, a lambda layer is used to compute the mean of these embeddings, and this mean will be sent to a softmax layer to output the most likely target word. The results will be compared with the actual target word by calculating the loss function, then doing back-propagation to adjust the weights in the embedding layer. This process will be repeated for all *(context, target)* pairs.

### Continuous Skip-gram

The Skip-gram architecture is similar to CBOW, but it predicts the surrounding window of context words by the target word. Here the weights of nearby context words are greater than the context words which are more distant.

Each training observation will have a pair of input words, one target word and one context word. If the pair of words has contextual meaning, we will label it as $Y = 1$, otherwise as $Y = 0$. Each input will be passed to the embedding layer of dimension *vocabulary size * embedding size* to produce word embeddings for these two words. Then, a merge layer is used to compute the dot product of these two embeddings. This value will then be sent to the sigmoid layer to generate the output which will be 0 or 1. The results will be compared with the actual label $Y$ by calculating the loss function, then back-propagating the errors to adjust the weights in the embedding layer. This process will be repeated for all *(context, target)* pairs.

### 2.2.2 GloVe

The Global Vectors for Word Representation, or GloVe is an extension to the Word2Vec technique for efficiently learning word vectors. It was invented by Pennington et al. [4] at Stanford in 2014.

Instead of using a window to determine local context, GloVe develops an explicit word co-occurrence matrix by using the statistics throughout the whole text corpus. Hence, GloVe model is trained on both the local context of words as well as the global statistics of the corpus and this makes the model capable of learning both the syntactic and semantic properties of words. While Word2Vec is a predictive model which learns vectors to improve the predictive ability, GloVe is a count-based model. A count-based model learns by reducing dimensionality on a co-occurrence counts matrix $X$.

In this model, the entries $X_{ij}$ denote the number of times word $j$ appears in the context of word $i$. Then, we denote $P_{ij} = P(j|i) = X_{ij}/X_i$ as the probability of word $j$ occurs in the context of word $i$, here $X_i$ is the number of how many words appear in the context of word $i$. For words $k$ related to word $i$ but not word $j$, the ratio $P_{ik}/P_{jk}$ will be large. Similarly, for words $k$ related to $j$ but not $i$, the ratio should be small. For words $k$ that are related to both $i$ and $j$, or not related to both $i$ and $j$, the ratio should be close to one. Compared to probabilities calculated from previous passage, the ratio computed here distinguishes relevant words from irrelevant words better, and it may also discriminate between the two relevant words.

Figure 2-3 illustrates the architecture of the GloVe model. The input consists of a one-hot encoded representation of a word. In this model, the word embedding matrices function as weight matrices, resulting in the output being a vector comprised of the inner products of the word vectors. The embedding matrices are updated based on the gradient of the loss function.

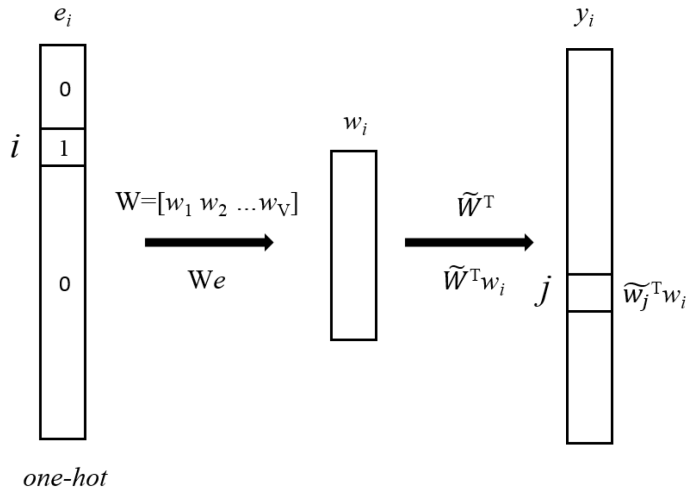


Figure 2-3: Model architecture of GloVe model
Adapted from [17]

The GloVe algorithm is based on the observation that the ratios of word co-occurrence probabilities can carry semantic meaning. In this algorithm, It involves a weighted least squares optimization to learn word vectors by minimizing the difference between the dot product of the word vectors and the logarithm of their co-occurrence probabilities. Hence, GloVe is trained by using gradient descent algorithm to minimize the loss function below:

$$L = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

Here,

- $w_i^T$ and $\tilde{w}_j$ are the word and context vector representations for words $i$ and $j$ respectively.
- $b_i$ and $\tilde{b}_j$ are the bias terms for words $i$ and $j$ respectively.
- $X_{ij}$ represents the number of times word $i$ occurs in the context of word $j$ in the corpus.
- $V$ is the number of unique words in the vocabulary.
- $f(X_{ij})$ is a weighting function, typically defined as:

$$f(x) = \begin{cases} (\frac{x}{x_{\max}})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

  where $x_{\max}$ and $\alpha$ are hyperparameters.

  The word embedding for Glove is:

$$v_i = w_i + \tilde{w}_i$$

The algorithm of GloVe iteratively updates the word vectors and bias terms by minimizing the loss function $L$. Once the optimization process is complete, the resulting word vectors can be used to perform various NLP tasks, such as semantic similarity measurement, analogy solving, or as input to downstream machine learning models.

Similarly, word embedding generated by GloVe model can be also interpreted by applying arithmetric operations or calculating the cosine similarity between word vectors. However, it is important to note that individual dimensions in the GloVe embeddings are not inherently interpretable. The primary focus of GloVe embeddings is to encode meaningful semantic relationships between words.

# Chapter 3

# Lie Groups

## 3.1 Overview of Lie Groups

**Definition: Lie Groups**

A (real) Lie group is a (real) smooth manifold $G$ which is also a group and such that the group product $G \times G \mapsto G$ and the inverse map $g \mapsto g^{-1}$ are smooth maps [7].

A manifold, on the other hand, is a topological space that looks locally like Euclidean space. In other words, around every point on the manifold, there is a neighborhood that is homeomorphic to an open set in Euclidean space. Two spaces are said to be homeomorphic if there exists a homeomorphism, which is a bijective and continuous function that maps one space onto the other and has a continuous inverse function. They are named after the mathematician Sophus Lie, who first studied them in the 19th century. Lie groups are used in many areas of mathematics, physics, and engineering, including in the study of symmetries, differential geometry, and number theory.

Examples of Lie Groups:

1. Circle group ($\mathbb{S}^1$): The circle group consists of complex numbers with an absolute value of 1, i.e., $\mathbb{S}^1 = z \in \mathbb{C} : |z| = 1$. It forms a Lie group under multiplication and can be visualized as a circle in the complex plane.

2. $\mathbb{R}^n$: $\mathbb{R}^n$, which is a set of all $n$-dimensional real vectors together with addition operation and neutral element 0 is a Lie group.

3. Rotation group ($SO(2, \mathbb{R})$): This group consists of all $2D$ rotations around the origin:
$$SO(2, \mathbb{R}) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} : \theta \in \mathbb{R}/2\pi\mathbb{Z} \right\}$$

It is isomorphic, i.e., there exists a one-to-one correspondence between their elements and operations, while maintaining the same properties and structure, to the circle group ($\mathbb{S}^1$) and can be visualized as a set of rotation matrices that keep the origin fixed.

Non-example of Lie Group:

1. Sphere ($\mathbb{S}^2$): The 2-sphere is the set of points in $\mathbb{R}^3$ equidistant from a given center, usually the origin. As a matter of fact, a Lie group is a manifold that is parallelizable, specifically, it possesses a non-vanishing vector field. However, Hairy ball theorem states that there is no non-vanishing tangent vector field on $\mathbb{S}^2$. Hence, $\mathbb{S}^2$ is not a Lie group.

> **Theorem: Hairy ball theorem**
>
> On even-dimensional $n$-spheres, a continuous tangent vector field that never vanishes does not exist [18].

## 3.2 Algebraic Structure and Dimensionality Analysis

### 3.2.1 Lie Algebra

A crucial approach to studying Lie groups involves exploring their corresponding Lie algebras. In this section, we will delve into the concept of a Lie algebra and the relationship between Lie algebras and Lie groups.

Consider a Lie group $G$ consisting of $n \times n$ matrices. Much of $G$'s structure can be comprehended by analyzing $G$'s behavior near the identity element. In mathematical terms, a smooth curve $A(t)$ in $G$ is examined with $A(0) = I$, and we seek the feasible values of $A'(0) = \frac{d}{dt}A(t)$ at $t = 0$. Essentially, we think of $G$ as a submanifold embedded in the vector space $\mathbb{R}_{n\times n}$ and look for the tangent space to $G$ at $I$. The Lie algebra $\mathfrak{g}$ that corresponds to the Lie group $G$ is defined as the tangent space to $G$ at $I$: it is the set of all possible values of $A'(0)$ when we consider all smooth curves $A(t)$ in $G$ satisfying $A(0) = I$. Since a tangent space to a manifold is always a vector space, the Lie algebra $\mathfrak{g}$ is also a vector space, making it easier to analyze than $G$, which is a nonlinear manifold. Hence, Lie algebras are introduced.

### 3.2.2 Dimensionality of Lie Groups

The dimensionality of a Lie group is usually found by computing the number of independent parameters needed to specify an element of the group. This can be done by considering the tangent space of the identity element of the Lie group, which is a vector space with a basis that corresponds to the generators of the group. The number of generators of the group is equal to the dimension of the Lie algebra, and the dimension of the Lie group is equal to the dimension of the Lie algebra.

The following are illustrations of how to find the dimension of specific Lie groups, with short definitions of each group.

### General Linear Group

The general linear group $GL(n, \mathbb{R})$ is the set of all $n \times n$ invertible matrices,

$$GL(n, \mathbb{R}) = \{A \in M_{n\times n}(\mathbb{R}) : A \text{ is invertible}\} = \{A \in M_{n\times n}(\mathbb{R}) : \det A \neq 0\}$$

The tangent Lie algebra at the identity matrix $I$ is the set of all $n \times n$ real matrices (not necessarily invertible). Each matrix element can be chosen independently, so the dimension of $GL(n, \mathbb{R})$ is $n^2$.

### Special Linear Group

The special linear group $SL(n, \mathbb{R})$ consists of all $n \times n$ matrices with real entries and determinant equal to 1:

$$SL(n, \mathbb{R}) = \{A \in GL(n, \mathbb{R}) : \det A = 1\}$$

Consider a curve in $SL(n, \mathbb{R})$ given by a matrix-valued function $A(t)$ such that $A(0) = I$ and $\det A(t) = 1$ for all $t$. Differentiating $\det A(t)$ with respect to $t$:

$$\frac{d}{dt} \det A(t) = \mathrm{Tr}(\mathbf{adj}(A(t)\frac{dA(t)}{dt}) \quad \text{(Jacobi's formula, adapted from [19])}$$

Since $\det A(t) = 1$, the adjugate matrix $\mathbf{adj}(A(t))$ is equal to the inverse matrix $A^{-1}(t)$:

$$\frac{d}{dt} \det A(t) = \mathrm{Tr}(A^{-1}(t)\frac{dA(t)}{dt})$$

When $t = 0$,

$$\mathrm{Tr}(A^{-1}(0) * A'(0)) = \mathrm{Tr}(I * A'(0)) = \mathrm{Tr}(A'(0))$$

Since $\det A(t) = 1$ for all $t$, the derivative $\frac{d}{dt} \det A(t) = 0$ for all $t$, therefore, $\mathrm{Tr}(A'(0)) = 0$.

Thus, the tangent space at the identity matrix $I$ consists of all $n \times n$ real matrices with trace 0. There are $n^2$ total elements in an $n \times n$ matrix, but since the trace must be 0, once we have determined the values of $n - 1$ diagonal elements, the $n$-th diagonal element is automatically determined to ensure the trace remains zero, hence, there are $n^2 - 1$ independent elements, so the dimension of $SL(n, \mathbb{R})$ is $n^2 - 1$.

### Orthogonal Group

The orthogonal group $O(n, \mathbb{R})$ consists of all $n \times n$ matrices with real entries satisfying $A^T A = I$:

$$O(n, \mathbb{R})) = \{A \in GL(n, \mathbb{R}) : A^T A = I\}$$

Consider a curve in $O(n, \mathbb{R})$ given by a matrix-valued function $A(t)$ such that $A(0) = I$ and $A(t)^T A(t) = I$ for all $t$. Differentiating the condition $A(t)^T A(t) = I$ with respect to $t$ and evaluating at $t = 0$, we get $A'(0)^T + A'(0) = 0$, where $A'(0)$ is the derivative of $A(t)$ at $t = 0$. This implies that the tangent space at the identity matrix $I$ consists of all $n \times n$ skew-symmetric real matrices ($A^T = -A$).

Since the lower triangular entries of any skew-symmetric matrix are determined by the entries above the diagonal, and the diagonal entries of skew-symmetric matrix must be zero, the independent entries are:

$$(n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2}$$

Thus, the dimension of $O(n, \mathbb{R})$ is $\frac{n(n-1)}{2}$.

### Special Orthogonal Group

The special orthogonal group $SO(n, \mathbb{R})$ consists of all $n \times n$ matrices with real entries satisfying $A^T A = I$ and $\det A = 1$:

$$SO(n, \mathbb{R}) := \{A \in O(n, \mathbb{R}) : \det A = 1\}$$

Since the condition ($A^T A = I$) is the same for both the orthogonal group and the special orthogonal group, and the determinant condition ($\pm 1$ for $O(n, \mathbb{R})$ and 1 for $SO(n, \mathbb{R})$) does not affect the tangent space calculation at the identity matrix $I$, the tangent space at the identity matrix $I$ is the same as for both groups. This means there are $\frac{n(n-1)}{2}$ independent elements in the special orthogonal group as well, so the dimension of $SO(n, \mathbb{R})$ is $\frac{n(n-1)}{2}$.

## Group of Three-dimensional Rotations

A special orthogonal group ($SO(3, \mathbb{R})$) represents the set of all possible rotations in three-dimensional Euclidean space preserving both the origin and the Euclidean distance between points [20], for all rotational matrices $R$,

$$SO(3, \mathbb{R}) := \{R \in M_{3\times 3}(\mathbb{R}) : R^T R = R R^T = I_3, \det R = 1\}$$

The $SO(3, \mathbb{R})$ group is a Lie group, hence it is a smooth and continuous group that is differentiable. This allows for the use of calculus and geometric methods in analyzing and controlling the behavior of systems in the group. Besides, $SO(3, \mathbb{R})$

group is a compact group, meaning that all its elements are bounded. The $SO(3,\mathbb{R})$ group represents the potential rotational symmetries and spatial orientations of an object. Its representations play a crucial role in physics, as they correspond to elementary particles with integer spin.

Various methods have been proposed for the representation of rotations in the $SO(3,\mathbb{R})$ group. The most widely used method is the Euler angle representation, which represents a rotation as a combination of three consecutive rotations about the $x$, $y$, and $z$ axes. Another popular method is the quaternion representation, which represents a rotation as a four-dimensional vector.

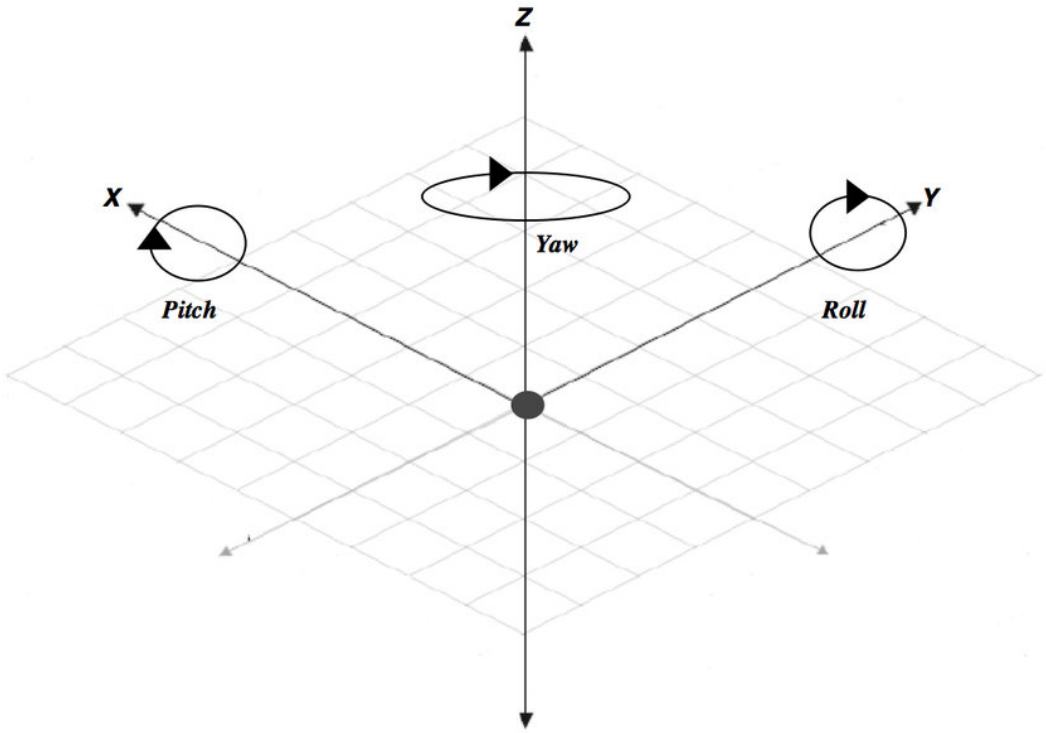


Figure 3-1: Yaw, pitch and roll about $x$, $y$ and $z$ axes
Adapted from [21]

# Chapter 4

# Factor Analysis

## 4.1 Overview

Factor analysis is mainly used to condense data in order to facilitate the interpretation and comprehension of relationships and patterns. It is based on the concept that observable and quantifiable variables can be simplified into a smaller set of latent variables that possess a shared variance and remain unobservable,this process is usually referred to as dimensionality reduction [22]. Although these unobservable factors are not directly measured, they can be used to represent various variables.

It is essential to ensure univariate and multivariate normality within the dataset when conducting a factor analysis [23]. Besides, the absence of univariate and multivariate outliers is crucial. One of the key assumptions in determining factors is the existence of a linear relationship between the factors and variables when calculating their correlations [24].

However, factor analysis has some limitations that should be considered when using it, such as multicollinearity, overfitting and issues with determining the number of factors to retain. Besides, the interpretation of the factors can be difficult because the factors are not directly observable. Finally, it's important to keep in mind that factor analysis assumes that the variables are continuous, which might restrict its application when dealing with categorical data.

## 4.2 Two Main Techniques of Factor Analysis

There are various types of factor analysis strategies, such as Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA).

### EFA

In an exploratory way, EFA is used to determine the underlying structure of a set of data. It is frequently used when researchers have no prior knowledge or assumptions

about the factor structure of the dataset. It is an unsupervised technique, therefore it may be used without a predetermined model or hypothesis to test. After extracting the factors by using EFA methods, the researchers will use factor rotation methods such as Varimax or Quartimax rotation to try to interpret the factors.

#### CFA

Contrarily, CFA is is a more rigid approach that tests a specific model or pre-specified hypothesis about the underlying structure of a set of variables based on prior knowledge. It is often used to compare various factor structures or to validate or evaluate the factor structure found in an EFA. Because CFA is a supervised approach, a specific model or hypothesis must be evaluated in order to use it. It estimates the model's parameters using statistical techniques like maximum likelihood estimation (MLE) or weighted least squares (WLS), and then assesses how well the model fits the data.

## 4.3 Theoretical Foundations of Factor Analysis

Prior to mentioning the key procedures of factor analysis, it is important to understand the theoretical concepts of factor analysis:

### 4.3.1 Mathematical Model

Below is the mathematical model of factor analysis:

$$\begin{aligned} X_1 &= a_{11}F_1 + a_{12}F_2 + \cdots + a_{1m}F_m + \epsilon_1 \\ X_2 &= a_{21}F_1 + a_{22}F_2 + \cdots + a_{2m}F_m + \epsilon_2 \\ &\vdots \\ X_p &= a_{p1}F_1 + a_{p2}F_2 + \cdots + a_{pm}F_m + \epsilon_p \end{aligned}$$

In this model, $p$ represents the number of variables $(X_1, X_2, \ldots, X_p)$, while $m$ denotes the number of underlying factors $(F_1, F_2, \ldots, F_m)$. The model assumes that there exist $m$ underlying factors, and the equation above shows how $X_j$ can be expressed in latent factors by a linear function combined with a residual value $\epsilon_j$. The goal for this model is to achieve the highest correlations.

Factor loadings are represented by $a_{j1}, a_{j2}, \ldots, a_{jm}$, where $a_{j1}$ indicates the factor loading of the $j$-th variable on the first factor. The residual value or error is symbolized by $\epsilon_j$. Factor loadings provide insights into the contribution of each variable to the factor; a larger factor loading signifies a greater contribution to that factor [25].

Factor analysis employs matrix algebra for its computations. Typically, a factor analysis using a correlation matrix generates standardized data, making it suitable for variables from different scales [8]. In contrast, a covariance matrix-based factor

analysis is applied to similar variables. The correlation matrix is often preferred due to its ease of interpretation compared to covariance tables.

The core concept of factor analysis is to seek factors such that, once extracted, no intercorrelations remain between any pairs $X_i$ and $X_j$ ($i \neq j$). In other words, all pairs of elements $(X_i, X_j, ..., X_p)$ are conditionally independent, given the values of $F_1, F_2, ..., F_m$.

### 4.3.2 Variance

Factor analysis uses variances, which is equivalent to the square of the factor loadings, to generate communalities between variables. Usually, the extraction objective is to reduce common variance in the first factor [23]. Communality estimates illustrate the shared information between a variable and all factors involved in the analysis. The communality ($h^2$) is the sum of the squared loadings:

$$h_j^2 = a_{j1}^2 + a_{j2}^2 + \cdots + a_{jm}^2$$

The computed communality indicates the percentage of a variable that can be predicted based on the factors' knowledge. Variables with low communalities (less than 0.20) are often excluded from the analysis.

In factor analysis, there is another type of variance, which is called unique variance ($u^2$), it represents the percentage of variance that does not include common factor variance, as shown by the formula:

$$u^2 = 1 - h^2$$

The total variance of a variable in factor analysis is composed of three components: communality, specificity, and unreliability. Communality represents the shared variance among variables, while specificity accounts for the unique variance of each variable. Unreliability refers to the measurement error in the variable.

## 4.4 Essential Elements of Factor Analysis

The two main components of factor analysis are factor extraction and factor rotation, as these steps help in identifying the underlying latent factors or structure in the data.

### 4.4.1 Factor Extraction

Factor extraction is the first step in factor analysis, the objective of this step is to identify a smaller number of unobserved factors that can explain the relationships among the observed variables. Various techniques can be used for factor extraction, and the two most common methods are Principal Component Analysis (PCA) and Common Factor Analysis (CFA). We will discuss both methods as follows:

**Principal Component Analysis**

PCA is a method which is used to transform the original variables into a new set of uncorrelated variables called principal components. The principal components are linear combinations of the original variables.

Before doing PCA, it is common to standardize the data, to make sure each variable has a mean of 0 and a standard deviation of $x$. This ensures that variables are comparable and there cannot exist any variance that can dominate the analysis. Next, PCA will compute the correlation matrix. In this matrix, the linear relationship between each pair of variables are measured. The diagonal elements represent the correlation of a variable with itself, hence they are 1s.

After that, PCA computes the eigenvalues and eigenvectors of the correlation matrix. Eigenvalues represent the amount of variance explained by each principal component, while eigenvectors contain the factor loadings. Based on the eigenvalues, components with higher values are chosen, as they explain more variance in the data. In practice, criterion like the Kaiser rule (eigenvalues $> 1$) or a scree plot (point of inflexion) is used to determine the number of components to retain.

**Common Factor Analysis**

CFA aims to identify latent factors that explain the shared variance among observed variables. While PCA is concerned with explaining the total variance in the data, on the other hand, CFA focuses on explaining the shared variance among variables due to the underlying factors.

In the first step of CFA, it starts by estimating the communality ($h^2$) for each variables. The correlation matrix's diagonal elements are replaced with these communality estimates. Then, The latent factors are extracted from the adjusted correlation matrix. Various techniques can be employed during this step, such as maximum likelihood estimation or principal axis factor. For example, maximum likelihood estimation is used to estimate the factor loadings, and this is achieved by maximizing a likelihood function which represents the probability of observing the data given the model parameters. In factor analysis, the likelihood function is the joint probability density function of the observed variables, which can be derived from the multivariate normal distribution.

Both PCA and CFA have their strengths and limitations, and the choice of method depends on the research question and assumptions about the underlying data structure. After factor extraction, the next step in factor analysis is factor rotation, which helps simplify the factor structure and make it more interpretable.

### 4.4.2 Factor Rotation

After doing factor extraction, the second main component of factor analysis is factor rotation. Factor rotation is essential in factor analysis as it can help to simplify the factor structure and make the results become more interpretable by maximizing high

loadings and minimizing low loadings for each factor. There are two types of factor rotation, such as orthogonal rotation and oblique rotation.

The total amount of variance explained by the factors will not be changed after doing the factor rotation step, even though it may merely redistribute the loadings to make interpretation of the factors easier. Besides, the choice of rotation method should be based on the theoretical knowledge of the constructs being measured and the forms of the relationships among the variables.

### Orthogonal Rotation

Orthogonal rotation assumes that the factors are uncorrelated, meaning the factors are statistically independent of each other. Two widely used orthogonal rotation methods are Varimax and Quartimax.

Varimax seeks to maximize the variance of the squared factor loadings within each factor, simplifying the factor structure by having high loadings for a small number of variables and low loadings for the remaining variables. This method helps in achieving a clear distinction between the factors and makes the interpretation of the factors easier.

Quartimax is another orthogonal rotation which is commonly used to minimize the number of factors required to explain a variable. Instead of simplifying the columns, it simplifies the rows of the loadings matrix. Quartimax often yields a general factor with loadings for numerous variables, therefore, this is similar as the factor which is unrotated. This method is advantageous when many variables are correlated, allowing for the emergence of a dominant factor.

### Oblique Rotation

Oblique rotation enables the factors to be correlated, hence the factors are not compulsory to be strictly independent of each other. Two popular oblique rotation methods are Oblimin and Promax.

The Oblimin rotation method seeks to simplify the factor structure by minimizing the cross products of factor loadings within factors, while allowing factors to be correlated, hence most of the factor loadings will become close to zero. Oblimin is a typical approach when seeking an oblique (non-orthogonal) solution in factor analysis.

On the other hand, Promax starts with an initial orthogonal rotation (often Varimax) and then adds a second step, which involves applying an oblique transformation to the orthogonally rotated solution to allow for correlation between factors. This method entails elevating the loadings to the fourth power, which ultimately leads to increased correlations among the factors and attains a simple structure. Besides, this method is occasionally used for extremely large datasets since Promax has advantages in computational time as compared with the Oblimin approach.

# Chapter 5

# Methodology

## 5.1 Data Preprocessing

The data was loaded into R, using "The Adventures of Sherlock Holmes" as the source material for the study [26]. During the preprocessing phase, the text was cleaned by converting all characters to lowercase and removing any non-character symbols. This process ensured a consistent and simplified format for subsequent analysis.

A Term Co-occurrence Matrix (TCM) was constructed with a skip-gram window size of three. To avoid an overly large matrix, the vocabulary was trimmed by retaining only words that appeared a minimum of five times. This step effectively eliminated uncommon words, streamlining the dataset for further processing. Additionally, stopwords were removed from the text to eliminate low-level information and emphasize the more important, relevant content.

Finally, the GloVe algorithm was applied to "The Adventures of Sherlock Holmes" dataset to generate word embedding vectors with dimensions $d = 20$. Note that the GloVe model is optimized by employing the gradient descent algorithm to minimize the loss function:

$$L = \sum_{i,k=1}^{V} f(X_{ik})(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik})^2$$

The gradient descent in the GloVe model is controlled by the following parameters:

1. Dimension $d$: This is the dimensionality of the word and context vectors, denoting that $w \in \mathbb{R}^d$ and $\tilde{w} \in \mathbb{R}^d$.

2. Maximum iterations: The maximum number of iterations after which the gradient descent process will be terminated.

3. Convergence tolerance $\epsilon$: Gradient descent will be stopped if the change in the loss function is less than $\epsilon$.

4. The threshold $x_{\max}$: This parameter influences the weighting function $f(X_{ik})$. When $x \geq x_{\max}$, $f(x)$ is equal to 1.

These parameters play a significant role in controlling the training process and performance of the GloVe model.

These vectors provided a comprehensive representation of the words' relationships within the text, contributing valuable insights into the underlying structure and thematic connections in the literary work.

## 5.2 Main Activity

The objective was to produce a word embedding matrix with interpretable entries while maintaining its semantic structure. To achieve this, the word embedding matrix $W$ was iteratively rotated using orthogonal rotation matrices $R$. Orthogonal rotation matrices were chosen to preserve the norm of word vectors and the angle between different word vectors, ensuring that the semantic structure remains intact.

Three loss functions were used to minimized, inspired by the concept of Varimax rotation in factor analysis and regularization techniques. In this study, Particle Swarm Optimization (PSO) algorithm was used to find $\frac{d(d-1)}{2}$ optimal parameters, based on the understanding that an orthogonal rotation matrix of dimension $d$ has $\frac{d(d-1)}{2}$ independent entries. Besides, to preserve the rotation matrix's orthogonality during iteration, the concept of the matrix exponential of any skew-symmetric matrix as an orthogonal rotation matrix was used. This exponential function was suitable for use in PSO as it is an onto function [27].

To evaluate the interpretability of the word embedding matrix, an accuracy metric was defined. It was assumed that the presence of more zeroes or values close to zero in the matrix would enhance interpretability, as these could be considered as neutral properties. Moreover, large absolute values could also be treated as positive or negative properties. By optimizing the loss functions, a word embedding matrix was generated that balanced both interpretability and the preservation of semantic structure.

### 5.2.1 Particle Swarm Optimization (PSO)

#### 5.2.1.1 Overview

Particle Swarm Optimization (PSO) is a nature-inspired optimization algorithm developed by James Kennedy and Russell Eberhart in 1995 [28]. As stated in the paper, they asserted that when a group of fish or birds move together, they can benefit from the experience from other members, in other words, although individual birds search for food in a seemingly random manner, the entire flock can share their findings, enabling the group to achieve the most successful hunt.

By simulating the motion of a flock of birds, each bird can be viewed as assisting us in discovering the optimal solution within a high-dimensional search space, with the best solution identified by the flock representing the optimal solution in that space. The solution that is found by this approach is considered as heuristic because it does not guarantee finding the correct global optimal solution. However, it is often that the solution obtained through PSO is reasonably close to the global optimum.

#### 5.2.1.2 Algorithms

At the beginning, a population of particles is randomly generated, where each particle has a position and velocity. The position represents a candidate solution in the search-space, while the velocity determines how the particle will move. Each particle also keeps track of its personal best position (pBest) found.

The fitness of each particle which is known as how is the performance of this solution, is evaluated using a problem-specific objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$. For each particle, if the current position has a better fitness than its previous pBest, the pBest is updated. Among all the personal best positions, the global best position (gBest) is identified.

During the process, each particle's velocity would be updated by using the following equation:

$$v_i(t+1) = w * v_i(t) + c_1 r_1(pBest_i - x_i(t)) + c_2 r_2(gBest - x_i(t))$$

And the position of each particle would be updated as:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Here, $v_i(t)$ is the velocity of the particle $i$ at iteration $t$, $x_i(t)$ is the position of the particle $i$ at iteration $t$, $w$ is the inertia weight, $c_1$ and $c_2$ are cognitive and social coefficients, $r_1$ and $r_2$ are random numbers uniformly distributed in $[0, 1]$, and $pBest_i$ and $gBest$ are the personal best positions and global best position.

The algorithm repeats the steps above for a predetermined number of iterations or until a stopping criterion is met, such as, the fitness value (objective value) converges, or a maximum number of iterations is reached.

The updated positions of particles gradually move towards the global best solution, resulting in the swarm converging on the optimal solution. PSO is simple to implement as compared with gradient descent. It has relatively few parameters to tune, and is effective for a wide range of optimization problems, such as continuous, discrete, and combinatorial optimization problems.

### 5.2.2 Orthogonal Rotations

#### 5.2.2.1 Determining Dimension

In the process of calculating the objective value, it was crucial to ensure that the orthogonal rotation matrix maintained its orthogonality after updating its entries. By preserving orthogonality, the semantic structure of the word embedding matrix remained unchanged while optimizing the objective function. The formula $d < -round(0.5*(1+\sqrt{1+8*length(x)}$ is used to determine the dimension $d$ of the skew-symmetric matrix $U$ based on the number of parameters to be optimized, denoted as $x$. Here, the matrix $U$ is an element of the Lie algebra associated with the special orthogonal group.

For an $d \times d$ skew-symmetric matrix, there are $\frac{d(d-1)}{2}$ independent elements. Since there are $x$ parameters which represent these independent entries, we solve for $d$ (the dimension of the skew-symmetric matrix) with the following equations:

$$\frac{d(d-1)}{2} = length(x)$$

We rearrange this equation to a quadratic equation in $d$:

$$d^2 - d - 2length(x) = 0$$

Using the quadratic formula, we find the value of $d$:

$$d = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{1 \pm \sqrt{1 + 8 * length(x)}}{2}$$

$$d = \frac{1 + \sqrt{1 + 8 * length(x)}}{2} \quad (d > 0)$$

#### 5.2.2.2 Perserving Orthogonality

Upon determining the dimension of matrix $U$, the upper triangular entries of $U$ were filled with $x$ independent parameters. Subsequently, $U$ was converted into a skew-symmetric matrix through the operation $U - U^t$.

To obtain the corresponding orthogonal rotation matrix, the matrix exponential of the skew-symmetric matrix was calculated. This transformation ensured that the matrix remained orthogonal throughout the process. Essentially, the matrix exponential function maps skew-symmetric matrices, which form the associated Lie algebra of the special orthogonal group, to orthogonal rotation matrices that belong to the special orthogonal group.

For a skew-symmetric matrix $A$ where $A^T = -A$ in $M_{n\times n}(\mathbb{R})$, it is essential to know that $R = e^A$ is a rotation matrix belongs to special orthogonal group $SO(n, \mathbb{R})$. We can check this by verifying with the two conditions:

1. Orthogonality: Since $R^T = e^{A^T} = e^{-A}$,

$$RR^T = e^A e^{-A} = I \quad \text{and} \quad R^T R = e^{-A} e^A = I,$$

hence $R$ is a orthogonal rotation matrix.

2. Determinant: Note that the trace of a skew symmetric matrix is 0. By using the fact that $\det e^A = e^{\mathrm{Tr}(A)}$, it can be easily checked that the determinant of $R$ is 1.

Conversely, for any rotation matrix $R$ belonging to $SO(n, \mathbb{R})$, there exists a skew-symmetric matrix $A$ such that $R = e^A$. This relationship can be concisely expressed by stating that the exponential map $\mathfrak{so}(n, \mathbb{R}) \mapsto SO(n, \mathbb{R})$ ($\mathfrak{g}$ is the Lie algebra associated to the Lie group $G$) is onto [27]. Furthermore, we can demonstrate the surjectivity of this exponential map by using the following corollary:

**Corollary**

If $G$ is a connected and compact matrix Lie group, the exponential map is a map from the Lie algebra $\mathfrak{g}$ of a Lie group $G$ to the group is surjective. [29]

This is a sufficient condition for the exponential map to be surjective. In fact, $SO(n, \mathbb{R})$ is both compact and connected [7].

### 5.2.3 Loss Functions

Before presenting the proposed objective functions in this study, it is helpful to first introduce the objective functions of Varimax and Quartimax rotations in factor analysis as there are similarities between their objectives and the loss functions proposed in this study.

Varimax rotation aims to maximize the sum of the variance of the squared loadings. This method results in factors that are easier to interpret, as it simplifies the factor structure by minimizing the number of variables with high loadings on each factor. For a factor loading matrix of dimensions $p \times m$, the Varimax objective function is given by:

$$\text{maximize} \quad \frac{1}{p}\sum_{j=1}^{m}\sum_{i=1}^{p}(a_{ij})^4 - \sum_{j=1}^{m}\sum_{i=1}^{p}(\frac{1}{p}(a_{ij})^2)^2$$

where $a_{ij}$ represents the factor loadings.

On the other hand, Quartimax rotation seeks to maximize the sum of all loadings to the power of 4. This approach reduces the number of factors needed to explain the variance in each variable. For a factor loading matrix of dimensions $p \times m$, the Quartimax objective function is given by:

$$\text{maximize} \quad \frac{1}{p} \sum_{j=1}^{m} \sum_{i=1}^{p} (a_{ij})^4$$

Both Varimax and Quartimax rotations serve as valuable points of reference for the proposed objective function in this study, as their objectives share the common goal of enhancing interpretability and simplicity.

Besides, in this study, the $l_1$-norm was used as a means to promote sparsity in the resulting solution. Consider a vector $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$, the corresponding $l_1$-norm is:

$$l_1(\beta) = \sum_{i=1}^{p} |\beta_i|$$

$l_1$-norm is usually used for regularization, such as LASSO regularization, due to its ability to promote sparsity by generating a higher proportion of zero-weight parameters [30]. By minimizing the $l_1$-norm, the solution exhibited an increased proportion of zero or near-zero values. This characteristic which is known as sparsity is beneficial for enhancing interpretability and simplifying the structure. On the other hand, it is important to note that the $l_2$ norm could not be used in this context, as orthogonal rotation inherently preserves the $l_2$ norm. Consequently, the $l_1$-norm was chosen as the most suitable option to encourage sparsity.

### 5.2.4 Accuracy Metrics

In this study, we developed an accuracy metric to evaluate the interpretability of the resulting word embedding matrix ($WR$). Our accuracy metric is designed to capture the proportion of elements in the word embedding matrix that are close to zero or have large values, with the objective of promoting a more interpretable and sparse representation of the word embeddings.

To calculate the accuracy metric, we first defined the percentage thresholds for zero and large values, setting them to $\alpha = 10\%$ and $\beta = 90\%$, respectively. Next, we computed the norm of each word vector (row) in the word embedding matrix:

$$\|WR_i\| = \sqrt{\sum_{j=1}^{d} (WR_{ij})^2}$$

Based on these norms, we determined the zero and large thresholds for each word vector by multiplying the norms with the corresponding percentage thresholds:

$$T_{zero_i} = \alpha \cdot \|WR_i\|$$

$$T_{large_i} = \beta \cdot \|WR_i\|$$

After obtaining the thresholds, we calculated the proportions of elements in the word embedding matrix that are within the zero and large thresholds. To do this, we divided the number of elements with absolute values less than or equal to the zero thresholds and greater than or equal to the large thresholds by the total number of elements in the matrix ($s * d$, where s is the number of words and d is the dimension of the word embeddings).

$$P_{zero} = \frac{\sum_{i=1}^{s} \sum_{j=1}^{d} \mathbb{1}(|WR_{ij}| \leq T_{zero_i})}{s \cdot d}$$

$$P_{large} = \frac{\sum_{i=1}^{s} \sum_{j=1}^{d} \mathbb{1}(|WR_{ij}| \geq T_{large_i})}{s \cdot d}$$

Finally, we combined the zero and large proportions to obtain our accuracy metric. A higher accuracy score indicates that the word embedding matrix has a higher proportion of elements close to zero or with large values, which is desirable for promoting interpretability in the word embeddings.

# Chapter 6

# Results and Discussion

In this chapter, the outcomes of using various objective functions are presented and examined. The focus is on illustrating how the accuracy metric evolves in response to modifications the selected objective functions. This study used three objective functions, inspired by Varimax and Quartimax rotations in factor analysis, as well as $l_1$-norm that promote sparsity. By comparing the results obtained from different objective functions, insights can be gained into their respective impacts on the interpretability of the word embedding matrix.

## 6.1 Choice of Words

Within the word embedding matrix, a total of $1,909$ words remained even after retaining only words that appeared a minimum of five times and removing the stopwords. The size of the matrix posed computational challenges and it made us hard to interpret the resulting word embedding matrix effectively. To facilitate ease of interpretation, a strategy was designed to select subsets of words for orthogonal rotations, followed by a comparison of the results and accuracy for different objective functions.

Each subset contained three words, each representing a different property: one with a positive attribute, one with a negative attribute, and one with a neutral attribute. This approach simplified the interpretation process and enabled an investigation into which columns of the word embedding matrix could successfully express the attributes of the words.

Below are five sets of words that we chose in the study:

1. (*far, near, cold*): This set comprises words representing distance (far, near) and an unrelated term (cold) to assess the matrix's ability to distinguish between spatial properties and unrelated concepts.

2. (*long, short, mystery*): In this set, the words long and short denote contrasting lengths or durations, while mystery serves as a neutral term.

3. (*man, woman, clock*): This set includes words representing different genders (man, woman) and an unrelated term (clock). The goal is to evaluate the matrix's performance in capturing gender distinctions while disregarding unrelated concepts.

4. (*happy, angry, orange*): In this set, happy and angry represent opposing emotional states, while orange serves as a neutral term. This set aims to assess the matrix's ability to distinguish between contrasting emotions and unrelated terms.

5. (*easy, difficult, colour*): This set features words describing contrasting levels of difficulty (easy, difficult) and a neutral term (colour).

The choice of these five sets of words was based on the objective of examining the performance and interpretability of the word embedding matrix when applied to words with distinct attributes. By conducting orthogonal rotations on these smaller, more manageable subsets of words, it was possible to effectively analyze and compare the performance of different objective functions. This approach not only reduced computational time, but also provided valuable insights into the interpretability of the word embedding matrix in capturing the essential attributes of the words.

In addition to the five sets of words, each containing three words, we expanded our selection criteria in this study to include one more set consisting of six words:

(*woman, wife, man, husband, clock, mystery*)

This set was designed to include two words with positive attributes, two words with negative attributes, and two words with neutral attributes. Our goal was to potentially extract any gender-related information from a single column of the resulting word embedding matrix.

## 6.2 Loss Functions

In this study, a word embedding matrix after orthogonal rotations, denoted as $WR$, was subjected to optimization using three distinct objective functions. The matrix $WR$ consists of $s$ rows and $d$ columns. $s$ represents the number of words, while $d$ represents the dimensions of word embeddings. Each function aimed to minimize a specific criterion, ultimately contributing to the interpretability of the resulting word embeddings. The following are the three objective functions used:

1. Varimax Method:

$$\text{minimize} \quad \sum_{j=1}^{d}\sum_{i=1}^{s}(\frac{1}{s}(WR)_{ij})^2)^2 - \frac{1}{s}\sum_{j=1}^{d}\sum_{i=1}^{s}((WR)_{ij})^4$$

   This objective function attempts to achieve high variance within each column, hence, this leads to a more interpretable solution, as each column tends to be dominated by a small number of words with high loadings.

2. Varimax + $l_1$-norm Method:

$$\text{minimize} \quad \sum_{j=1}^{d}\sum_{i=1}^{s}(\frac{1}{s}(WR)_{ij})^2)^2 - \frac{1}{s}\sum_{j=1}^{d}\sum_{i=1}^{s}((WR)_{ij})^4 + \frac{1}{s}\sum_{j=1}^{d}\sum_{i=1}^{s}|(WR)_{ij}|$$

   The inclusion of the $l_1$-norm encourages sparsity by promoting the use of fewer non-zero elements in the word embedding matrix ($WR$). The idea of using this objective function is to maintain the interpretability of the word embedding matrix while also promoting sparsity.

3. $l_1$-norm Method:

$$\text{minimize} \quad \frac{1}{s}\sum_{j=1}^{d}\sum_{i=1}^{s}|(WR)_{ij}|$$

   This objective function solely focuses on minimizing the sum of the absolute values of the entries in the word embedding matrix ($WR$), it is used to promote sparsity by encouraging many entries to be zero or close to zero.

## 6.3 Results

### 6.3.1 Table of Accuracy

| | Accuracy | | |
|---|---|---|---|
| | Varimax | Varimax + $l_1$ | $l_1$ |
| far, near, cold | 0.8 | 0.73 | 0.7 |
| long, short, mystery | 0.72 | 0.7 | 0.7 |
| man, woman, clock | 0.83 | 0.62 | 0.63 |
| happy, angry, orange | 0.63 | 0.63 | 0.6 |
| easy, difficult, colour | 0.82 | 0.62 | 0.62 |

Table 6-1: Accuracy summary for different sets of words when using different loss functions

Table 6-1 above presents a summary of the accuracy achieved for five sets of words when using various loss functions. The objective functions compared include Varimax, Varimax combined with the $l_1$-norm, and the $l_1$-norm alone. The table illustrates the performance of these methods on five distinct sets of words, each containing a positive, negative, and neutral property.

In conclusion, the Varimax method tends to perform well due to its focus on maximizing the variance of the squared entries, which results in a clearer separation of features and improved interpretability, while the $l_1$-norm focuses on promoting sparsity but may not always capture the underlying structure effectively. The combined Varimax + $l_1$-norm method attempts to enhance the interpretability and sparsity, resulting in accuracy that is generally similar to either the Varimax or the $l_1$-norm alone.

### 6.3.2 Interpretation

| | V2 |
|---|---|
| far | 0.66 |
| near | -0.01 |
| cold | -0.80 |

(a) Before

| | V2 |
|---|---|
| far | 1.17 |
| near | -0.2 |
| cold | -0.03 |

(b) Varimax

Table 6-2: These tables present data extracted from the word embedding matrix before and after applying orthogonal rotations using Varimax objective function for (*far, near, cold*)

Based on the table 6-2, we can analyze the positivity, negativity, and neutrality of the words in relation to the V2 column.

The given values in the V2 column of the word embedding matrix provide insightful information about the association of the words "far," "near," and "cold" with a specific attribute. With a value of 1.17, "far" demonstrates a positive

association with the attribute represented by the V2 column. Conversely, "near" has a negative association with this attribute, as indicated by a value of $-0.2$. Meanwhile, "cold" has a slight negative association with the attribute, evidenced by a value of $-0.03$. However, this value is close to zero, suggesting that "cold" possesses a relatively neutral property in relation to the attribute.

Based on these values, it can be concluded that "far" exhibits a positive property, "near" displays a negative property, and "cold" maintains a neutral property represented by the V2 column. Therefore, it can be claimed that this column may be used to express the concept of remoteness.

| | V12 | V15 |
|---|---|---|
| happy | -0.12 | -0.64 |
| angry | -0.02 | 0.08 |
| orange | -0.09 | -0.96 |

(a) Before

| | V15 |
|---|---|
| happy | -0.98 |
| angry | 0.57 |
| orange | -0.02 |

(b) Varimax

| | V12 |
|---|---|
| happy | 1.00 |
| angry | -0.55 |
| orange | 0.08 |

(c) Varimax + $l_1$-norm

Table 6-3: These tables present data extracted from the word embedding matrix before and after applying orthogonal rotations using two distinct objective functions for (*happy, angry, orange*)

Similarly, referring to table 6-3, it can be concluded that the V12 column can be used to represent the concept of happiness. Besides, the V15 column is capable of expressing the sentiment of anger. These interpretations are based on the associations between the words and the attributes represented by the respective columns in table 6-3.

| | V8 |
|---|---|
| woman | 1.10 |
| wife | -0.04 |
| man | -0.19 |
| husband | -0.53 |
| clock | 0.12 |
| mystery | 0.37 |

(a) Before

| | V8 |
|---|---|
| woman | -0.74 |
| wife | -0.34 |
| man | 0.13 |
| husband | 0.71 |
| clock | 0.10 |
| mystery | -0.10 |

(b) Varimax

Table 6-4: These tables present data extracted from the word embedding matrix before and after applying orthogonal rotations using three distinct objective functions for (*woman, wife, man, husband, clock, mystery*)

In Table 6-4, it is evident that column V8 can be used to represent gender identity information as it captures some information about the gender-related concepts. This observation is based on the fact that the words "man" and "husband" have positive values, "woman" and "wife" have negative values, and "clock" and "mystery" have values close to zero. With this information, we can effectively identify words that exhibit positive, negative, or neutral attributes, respectively.

In brief, the Varimax method appears to perform the best among the compared methods due to its objective function which effectively provides a more distinct separation of features and improved interpretability. On the other hand, the $l_1$-norm primarily focuses on promoting sparsity but may not consistently capture the underlying structure in an effective manner. The Varimax + $l_1$-norm method seeks to enhance both interpretability and sparsity, resulting in an accuracy that generally similar to either the Varimax method or the $l_1$-norm method individually. By examining the rotated word embedding matrix, valuable interpretations can be drawn, such as identifying specific columns that can be used to express distinct concepts. Nonetheless, the interpretability of the resulting word embedding matrix becomes more challenging when the set of words includes a larger number of words.

# Chapter 7

# Conclusion and Future Work

## 7.1 Summary

Word embedding is a technique used in NLP, which is widely applied across various domains. However, the interpretability of word embeddings is limited, as individual elements of word vectors do not have distinct meanings. Recent research has sought to create interpretable word embeddings by integrating explicit knowledge, such as semantic lexicons or supervised categories into the word embedding process [5, 6]. Although these approaches have demonstrated significant improvements in word embedding interpretability, studies focusing on word embeddings with interpretable entries remain insufficient.

In this study, we proposed a method that rotated the word embedding matrix to minimize certain loss functions using the PSO algorithm to achieve a word embedding matrix with interpretable entries while preserving the orthogonality of the rotation matrix during iterations by using the principles of Lie group. Our findings indicated that the Varimax method performed best in our study, achieving the highest accuracy for various sets of words, and some word embedding matrices can be interpreted in some senses successfully.

## 7.2 Areas of Improvement

Despite attempts to achieve convergence, the algorithm did not converge even after reaching 10,000 iterations, which is the maximum number of iterations set for the PSO algorithm. The computations took a significant amount of time even when applied to smaller sets of words and reduced dimensions. Hence, this highlights the challenges associated with optimizing and efficiently processing large-scale word embeddings while maintaining interpretability.

## 7.3 Future Works

Gradient descent can be used as an optimization technique to enhance results and accelerate computational speed, potentially leading to more efficient and effective convergence. In this study, we were limited to only 20 dimensions due to computational speed of the algorithm. Exploring higher dimensions might reveal additional insights and improve the interpretability of the word embedding matrix. Furthermore, developing better objective function will be beneficial, as it may not only enable each individual column of the word embedding matrix to express specific attributes more accurately, but also make the convergence speed faster. These improvements can potentially lead to a better method to generate word embedding with interpretable entries.

# References


[1] K. R. Chowdhary. "Natural Language Processing". In: *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, 2020, pp. 603–649. ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19. URL: https://doi.org/10.1007/978-81-322-3972-7_19.

[2] Rémi Philippe Lebret. "Word Embeddings for Natural Language Processing". In: (2016), p. 174. DOI: https://doi.org/10.5075/epfl-thesis-7148. URL: http://infoscience.epfl.ch/record/221430.

[3] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013.

[4] 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)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[5] Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. "Ultradense Word Embeddings by Orthogonal Transformation". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 767–777. DOI: 10.18653/v1/N16-1091. URL: https://aclanthology.org/N16-1091.

[6] Lütfi Kerem Şenel et al. "Semantic Structure and Interpretability of Word Embeddings". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10 (2018), pp. 1769–1779. DOI: 10.1109/TASLP.2018.2837384.

[7] Alistair Savage. "Introduction to Lie groups". In: *Course Notes of MAT* 1411 (2015).

[8] An Gie Yong, Sean Pearce, et al. "A beginner's guide to factor analysis: Focusing on exploratory factor analysis". In: *Tutorials in quantitative methods for psychology* 9.2 (2013), pp. 79–94.

[9] Finale Doshi-Velez and Been Kim. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. DOI: 10.48550/ARXIV.1702.08608. URL: https://arxiv.org/abs/1702.08608.

[10] Yu Zhang et al. "A Survey on Neural Network Interpretability". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5 (2021), pp. 726–742. DOI: 10.1109/TETCI.2021.3100641.

[11] Gintare Karolina Dziugaite, Shai Ben-David, and Daniel M. Roy. *Enforcing Interpretability and its Statistical Impacts: Trade-offs between Accuracy and Interpretability*. 2020. DOI: 10.48550/ARXIV.2010.13764. URL: https://arxiv.org/abs/2010.13764.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[13] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[14] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[15] Anthony J. Myles et al. "An introduction to decision tree modeling". In: *Journal of Chemometrics* 18.6 (2004), pp. 275–285. DOI: https://doi.org/10.1002/cem.873. eprint: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/pdf/10.1002/cem.873. URL: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/cem.873.

[16] Lin Liu et al. "Towards an Efficient Privacy-Preserving Decision Tree Evaluation Service in the Internet of Things". In: *Symmetry* 12 (Jan. 2020), p. 103. DOI: 10.3390/sym12010103.

[17] Chang Liu et al. "Semantic Features Based N-Best Rescoring Methods for Automatic Speech Recognition". In: *Applied Sciences* 9.23 (2019). ISSN: 2076-3417. DOI: 10.3390/app9235053. URL: https://www.mdpi.com/2076-3417/9/23/5053.

[18] Keith Burns and Marian Gidea. *Differential Geometry and Topology: With a view to dynamical systems*. CRC Press, 2005.

[19] Jan R Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

[20] Hashim A. Hashim. *Special Orthogonal Group SO(3), Euler Angles, Angle-axis, Rodriguez Vector and Unit-Quaternion: Overview, Mapping and Challenges*. 2019. DOI: 10.48550/ARXIV.1909.06669. URL: https://arxiv.org/abs/1909.06669.

[21] Sophie Hall, Fridolin Wild, and Tjeerd Scheper. "Real-Time Auditory Biofeedback System for Learning a Novel Arm Trajectory: A Usability Study". In: Nov. 2019, pp. 385–409. ISBN: 978-3-319-64300-7. DOI: 10.1007/978-3-319-64301-4_18.

[22] David J Bartholomew, Martin Knott, and Irini Moustaki. *Latent variable models and factor analysis: A unified approach*. John Wiley & Sons, 2011.

[23] Dennis Child. *The essentials of factor analysis*. A&C Black, 2006.

[24] Richard L Gorsuch. *Factor analysis*. Psychology press, 2013.

[25] Harry H Harman and Harry Horace Harman. *Modern factor analysis*. University of Chicago press, 1976.

[26] Arthur Conan Doyle. *The Adventures of Sherlock Holmes*. George Newnes Ltd., 1892.

[27] Theodor Bröcker and Tammo Tom Dieck. *Representations of compact Lie groups*. Vol. 98. Springer Science & Business Media, 2013.

[28] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[29] Brian Hall. "Lie Groups, Lie Algebras, and Representations". In: *Graduate Texts in Mathematics* 222 (Jan. 2015). DOI: 10.1007/978-3-319-13467-3.

[30] Sanjiban Sekhar Roy et al. "Stock market forecasting using LASSO linear regression model". In: *Afro-European Conference for Industrial Advancement: Proceedings of the First International Afro-European Conference for Industrial Advancement AECIA 2014*. Springer. 2015, pp. 371–381.

# Appendix A

# Appendix Tables

**Table A-1**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| cold | 0.72 | 0.66 | -0.35 | 0.11 | 0.84 | 0.58 | -0.51 | 0.01 | -0.06 | -1.01 |
| far | -0.10 | -0.01 | -0.40 | -0.18 | 0.25 | -0.84 | -0.48 | 0.16 | 0.12 | -0.41 |
| near | 0.40 | -0.80 | -0.41 | 0.24 | 0.11 | 0.92 | -0.56 | 0.25 | -0.12 | 0.31 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| cold | 0.19 | -0.26 | 0.66 | -0.98 | 0.41 | 0.45 | -0.34 | -1.08 | 0.17 | -0.27 |
| far | -0.52 | -0.09 | 0.04 | 0.51 | 0.41 | -0.16 | 0.61 | 0.13 | -0.39 | 0.29 |
| near | 0.48 | 0.45 | 0.33 | 0.27 | 0.38 | 0.01 | 0.46 | -0.27 | -0.79 | 0.09 |

Table A-1: This is the word embedding matrix of (*far, near, cold*)

**Table A-2**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| cold | -0.02 | -0.03 | 0.02 | 0.06 | -0.01 | -0.04 | -0.01 | -0.04 | 0.04 | -0.02 |
| far | -0.08 | 1.17 | -0.15 | 0.12 | 0.77 | 0.14 | 0.10 | 0.16 | 0.14 | -0.20 |
| near | 0.07 | -0.20 | 0.08 | 1.92 | 0.28 | 0.09 | -0.13 | 0.08 | 0.15 | 0.10 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| cold | -0.06 | 0.08 | -0.01 | -0.00 | -0.04 | 0.04 | -2.58 | -0.03 | 0.07 | 0.01 |
| far | 0.10 | -0.07 | 0.28 | 0.44 | 0.36 | -0.44 | 0.12 | -0.05 | 0.05 | -0.19 |
| near | 0.05 | -0.22 | 0.22 | 0.02 | -0.31 | -0.10 | -0.19 | 0.09 | 0.05 | 0.16 |

Table A-2: This is the word embedding matrix after rotating it using the Varimax objective function for (*far, near, cold*)

**Table A-3**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| happy | -0.14 | -0.06 | 0.15 | -0.15 | 0.21 | 0.50 | 0.44 | 0.19 | 0.22 | 0.62 |
| angry | 0.38 | 0.56 | 0.04 | 0.59 | -0.36 | -0.19 | -0.29 | 0.22 | 0.03 | 0.15 |
| orange | 0.80 | -0.94 | -1.19 | -0.03 | -0.00 | 0.19 | 0.06 | -0.22 | -0.07 | 0.34 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| happy | -0.14 | -0.12 | 0.26 | -0.61 | -0.64 | 0.49 | -0.29 | -0.12 | -0.00 | 0.29 |
| angry | -0.10 | -0.02 | -0.45 | 0.18 | 0.08 | -0.08 | -0.04 | 0.18 | -0.09 | -0.34 |
| orange | 0.25 | -0.09 | 0.48 | 0.12 | -0.96 | -0.26 | 1.03 | 0.54 | -0.93 | -0.31 |

Table A-3: This is the word embedding matrix of (*happy, angry, orange*)

**Table A-4**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| happy | -0.00 | 0.03 | -0.13 | -0.26 | 0.01 | -0.12 | 0.26 | -0.02 | -0.14 | 0.72 |
| angry | 0.45 | -0.16 | -0.02 | 0.41 | -0.20 | -0.13 | 0.39 | 0.21 | -0.08 | -0.00 |
| orange | -0.02 | 2.58 | -0.10 | 0.04 | 0.07 | -0.05 | 0.07 | 0.06 | -0.06 | -0.00 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| happy | 0.22 | 0.04 | -0.60 | -0.30 | -0.98 | -0.15 | -0.33 | 0.23 | 0.06 | -0.12 |
| angry | -0.45 | -0.06 | -0.12 | -0.20 | 0.57 | 0.10 | 0.37 | 0.17 | -0.32 | -0.02 |
| orange | -0.04 | 0.13 | 0.02 | -0.03 | -0.02 | -0.03 | -0.01 | 0.06 | -0.03 | -0.00 |

Table A-4: This is the word embedding matrix after rotating it using the Varimax objective function for (*happy, angry, orange*)

**Table A-5**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| happy | 0.26 | -0.51 | 0.65 | -0.14 | -0.09 | -0.06 | 0.27 | 0.15 | 0.21 | 0.03 |
| angry | -0.07 | 0.06 | -0.01 | 0.10 | -0.19 | 0.47 | 0.01 | 0.04 | 0.14 | -0.10 |
| orange | -0.04 | -0.12 | -0.04 | 0.04 | 0.00 | 0.04 | -0.10 | -0.04 | -0.06 | 2.58 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| happy | -0.31 | 1.00 | -0.07 | -0.26 | 0.02 | -0.24 | 0.07 | 0.25 | -0.03 | 0.39 |
| angry | 0.29 | -0.55 | -0.23 | -0.27 | 0.55 | 0.34 | 0.38 | 0.09 | 0.36 | 0.01 |
| orange | -0.04 | 0.08 | -0.02 | 0.03 | -0.04 | 0.02 | -0.02 | 0.04 | -0.01 | -0.00 |

Table A-5: This is the word embedding matrix after rotating it using the Varimax + $l_1$ objective function for (*happy, angry, orange*)

**Table A-6**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| woman | -0.27 | 0.24 | -0.25 | -0.40 | 0.01 | -0.41 | 0.03 | 1.10 | 0.10 | -0.75 |
| wife | -0.02 | -0.09 | 0.26 | -0.07 | -1.27 | 0.78 | -0.33 | -0.04 | -0.03 | 0.10 |
| man | 1.09 | -1.17 | -0.78 | -0.18 | -0.22 | -0.41 | 0.21 | -0.19 | -0.76 | -0.16 |
| husband | -0.19 | -0.17 | -0.16 | -0.63 | -0.27 | 0.35 | -0.56 | -0.53 | 0.71 | -0.54 |
| clock | -0.10 | -0.32 | -0.27 | -0.13 | -0.63 | 0.59 | -0.01 | 0.12 | 0.23 | 0.64 |
| mystery | -0.50 | 0.05 | 0.01 | -0.27 | 0.55 | 0.21 | 0.24 | 0.37 | -0.65 | 0.25 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| woman | 0.18 | -0.22 | -0.56 | 0.15 | 0.14 | -0.60 | -0.34 | -0.20 | 0.70 | 1.09 |
| wife | -0.44 | -0.25 | -0.31 | -0.80 | 0.34 | 0.81 | -0.55 | -0.32 | 0.67 | 0.28 |
| man | 0.04 | 0.64 | -0.16 | -0.99 | 0.40 | 0.09 | -0.04 | -0.84 | 0.41 | 0.11 |
| husband | 0.32 | -0.40 | 0.01 | 0.38 | -0.56 | -0.20 | 0.83 | -0.41 | -0.22 | -0.37 |
| clock | -0.18 | -0.41 | -0.19 | 0.17 | 0.31 | -0.41 | 0.23 | -0.90 | 0.00 | -0.32 |
| mystery | 0.47 | -0.13 | 0.73 | 0.06 | -0.03 | -0.64 | 0.32 | 0.43 | 0.17 | -0.04 |

Table A-6: This is the word embedding matrix of (*woman, wife, man, husband, clock, mystery*)

**Table A-7**

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 |
|---|---|---|---|---|---|---|---|---|---|---|
| woman | 0.08 | -0.17 | -0.11 | -0.40 | -0.43 | 0.66 | 0.92 | -0.74 | 0.21 | -0.74 |
| wife | 0.09 | 0.31 | 0.01 | 0.08 | 0.09 | -0.47 | 0.09 | -0.34 | -0.07 | 0.21 |
| man | -0.08 | 0.72 | 0.27 | 0.18 | 0.02 | -0.29 | 0.21 | 0.13 | 0.22 | -0.07 |
| husband | 0.06 | -0.26 | 0.55 | -0.30 | 0.02 | 0.23 | -0.46 | 0.71 | 0.38 | -0.03 |
| clock | -0.19 | -0.07 | 0.30 | 0.56 | -0.15 | 0.53 | 0.37 | 0.10 | 0.29 | 0.53 |
| mystery | -0.43 | -0.44 | -0.32 | 0.05 | 0.37 | 0.23 | 0.36 | -0.10 | -0.20 | -0.41 |
| | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 |
| woman | 0.03 | 0.61 | 0.14 | 0.23 | -1.06 | -0.21 | 0.54 | -0.27 | 0.05 | 0.44 |
| wife | 0.03 | 0.25 | 0.34 | -0.15 | -0.07 | -0.11 | 0.38 | -0.25 | -2.04 | -0.16 |
| man | 0.54 | -0.10 | 2.18 | -0.12 | -0.07 | -0.25 | -0.29 | -0.72 | -0.04 | -0.12 |
| husband | 0.10 | -0.09 | -0.13 | -0.34 | -0.40 | -0.72 | -0.00 | 1.27 | -0.13 | 0.15 |
| clock | 0.40 | 0.22 | 0.07 | -0.13 | 0.39 | -0.87 | -0.06 | 0.30 | -0.63 | -0.24 |
| mystery | -0.38 | 0.63 | -0.01 | 0.30 | 0.25 | 0.22 | -0.48 | 0.50 | 0.64 | -0.44 |

Table A-7: This is the word embedding matrix after rotating it using the Varimax objective function for (*woman, wife, man, husband, clock, mystery*)