Statistical Relational Learning for Knowledge Graphs

Link Prediction with Latent Feature Models

Luyolo Magangane

Department of Mathamatics
Stellenbosch University

This dissertation is submitted for the degree of
Master of Sciences

July 2019

I would like to dedicate this thesis to my loving parents . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Luyolo Magangane
July 2019

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Statistical Relational Learning

1.1.1 Artificial General Intelligence

The ultimate ambition of artificial intelligence (A.I.) research is to produce human-level intelligence. Such intelligence can take form the form of conversational artificial general intelligence (AGI). The problem can be posed as a question and answer game [Alan Turing]. The premise - a human asks a series of questions to an unobservable agent, where the task is for the agent to provide answers indistinguishable from human responses. The challenge of course is the breadth of topics that can be queried, as well as the permutations of possible coherent responses.

1.1.2 General Question Answering

This task is also known as general question answering (GQA) [references]. Typical A.I. implementations operate within well bounded, narrow domains. For example audio to text synthesisers [references]; object detection within images and videos [references]; financial, meteorological and operational forecasting [references]. All of these modelling applications focus more generally on finite perceptive domains [references]. In order to achieve AGI, reasoning capability is also required [requirements for AGI reference].

1.1.3 Applications of General Question Answering

Intelligent virtual assistants (IVA) such as Alexa, Cortana, Google Assistant and Siri are mobile application attempts at AGI; Watson, Wolfram Alpha and Knowledge Engine are other GQA computer systems. So often a user will ask one of these systems a question which results in a search engine lookup, due to failure in comprehending the question and the capability of providing a coherent response.

1.1.4 General Question Answering Approaches

Reinforcement learning. Will not be covered in this dissertation.... Knowledge representation and reasoning (KRR) is an active area of research being used as an approach to solve this problem [references]. The field aims to use mathematical logic to construct knowledge representations about a domain using knowledge graphs. Knowledge graphs (KG) model information in the form of facts presented as subject-predicate-object triples. These triples are expressed as entity-relational sets (e_1, r, e_2) where e_1 is the subject entity and e_2 is the object entity, and the relationships, r, between them. KGs have been used for information extraction, search query augmentation and general question answering. Google Knowledge Graph, DBPedia and Yago are some of the largest graph-structured database knowledge graphs implemented as knowledge bases.
KGs can be used to generate new knowledge from existing facts. These KGs can then be queried to provide logically consistent answers, more formally ontological reasoning is used to query ontological domains.

1.1.5 Uncertainty in Knowledge Representations

The problem with KRR is it contains no mechanism of capturing the uncertainty in known facts, as well as the uncertainty when new facts are generated, statistical relation learning (SRL) is an approach to solve this problem and active research area. SRL comprises three primary tecnhiques: Latent Feature Modelling - this comprises building entity-rational representations for relationship classification, Graph Feature modelling - Using graphical structures to model entity-relational properties within a domain, and Inductive Probabilistic Logic programing - Building probabilistic models of facts to directly model uncertainty in knowledge bases. Tasks in SRL include link prediction, entity-resolution and link-based clustering. Link prediction .... Entity-resolution ..., Link-based clustering ... Link prediction with Latent Feature Models is the focus of this dissertation.

A *LaTeX class file* is a file, which holds style information for a particular LaTeX.

$$CIF: \quad F_0^j(a) = \frac{1}{2\pi\iota} \oint_\gamma \frac{F_0^j(z)}{z-a} dz \tag{1.1}$$

1.2 Modelling Techniques

1.2.1 Latent Feature Modelling

Entities and relations are words that can be represented as real-valued vectors [references]. These real-valued vectors form part of a euclidean embedding space that represents a knowledge domain [references]. The entity and relational vectors can be randomly generated, or be pre-computed to capture semantic meaning [references]. A classification model can then be constructed that generates a probability distribution over probable facts within the knowledge domain. In order to compute the probability distribution, a number of latent feature modelling techniques are used, including tensor factorisation [references], circular correlations [reference] and convolutional feature maps [reference]. These methods can broadly be defined as linear and nonlinear. Attractive attributes of linear latent models are their simplicity, ease of implementation and computational efficiency. Linear latent models however suffer from a lack of expressiveness and struggle to model complex, contradictory or incoherent relationships between entities. Nonlinear latent feature models are able to produce more expressive latent feature sets, and so more adept at capturing complex relationships. Nonlinear models however suffer from computational inefficiency and poorly generalise concepts.

1.2.2 Graph Feature Modelling

In Graph Feature Modelling, Knowledge Graphs (KG) are used to model domains. KGs are composed of nodes and edges, where nodes represent entities and edges represent relations. The graphical structure then captures local, quasi-local and global domain properties. This global structure exhibits particular properties about relations within the domain, characteristics of the entities of the domain, and local entity-relational sub-structures. These graph structure properties are used in supervised [reference] and unsupervised [Graph Infomax] settings for SRL tasks such as link prediction and entity-resolution. The directional nature of edges in graph structures (uni-relational and bi-relational) is also exploited to further enhance the fulfillment of SRL tasks [reference]. The assumption in general in KGs is that similar entities will be collocated within a local and quasi-local regions, and that global similiarty patterns between entities will be captured by the ensemble of all paths between entities. Link-based clustering [reference] is thus used at all these structural scopes, and supports link prediction and entity-resolution tasks.

1.2.3 Inductive Probabilistic Logic Programming

Inductive logic programming uses ontological facts to discover new facts within a knowledge domain [reference]. logical rules check for things such as consistency. coherence and contradiction. Knowledge domains are implemented as knowledge bases (KB) that follow the resource description framework [reference]. KBs initialised in two steps: fact recording and materialisation - the discovery of new facts by running logical queries over the entire KB. KBs are extremely computationally demanding [reference], they also suffer from an inflexibility in modelling complex relationships due to their exactness, a fact is either true or false with no measure of ambiguity. Probabilistic logic programming languages have recently gained a lot of attention as flexible alternatives to logic programming languages as they are able to capture uncertainty in logical assertions through by modelling probability distributions over KB facts using stochastic variational inference. These models are thus more flexible in modelling complex relationships, and are also more computationally efficient [reference]. Inductive probabilistic logic programming has recently gained a lot of research attention due to it's capability of extending probabilistic logic programming languages with the capability of knowledge discovery.

1.3 Link Prediction with Latent Feature Models

1.3.1 Knowledge Graph Latent Feature Models

Link prediction with latent feature models involves building entity-relational representations from the nodes and edges of knowledge graphs expressed as subject-predicate object triples. These triples explicitly model facts within a knowledge domain. Entity and relational representations are commonly implemented as real-valued vectors. The vectors are then combined using compositional models, such as neural networks, to produce latent relational representations that can be used to compute the likelihood of plausible relationships between entities. The domain can be said to represent a multidimensiona embedding space into which the entities and relations are projected. Knowledge graph based latent feature modelling approaches are similar to semantic embedding representations [rerferenc]. They differ in that knowledge graph approaches explicitly model entity-relational interactions, and semantic embedding approaches rely on the distributional word representation techniques [reference], relying on Skip-Gram [reference] and Contious Bag of Words [reference] to generate word representations. This is an implicit modelling of relationships between word vectors.

1.3.2 Factorisation of Latent Feature Models

Factorisation attempts to model concepts between words, these facts are discovered using unsupervised techniques such as singular value decomposition. In the case of knowledge graphs, we obtain explicit representations of these concepts and can use them for entity-relational transformations that represent intermediate relational concepts that can then be used to determine plausible relationships when tested against subject entities. Tensor factorisation is an approach used for link prediction with latent feature models. It involves modelling entity relationships as matrix slices that comprise a relational tensor. The entity between entities is then computed using a bilinlear tensor product [reference], where the inner product of the object entity is taken with the matrix relational representation before an inner product of the resultant representation is taken with the subject entity. Bilinear tensor factorisation models are efficient in their number of parameters but lack expressiveness. Multilayer perceptrons have been used to overcome the lack of expressiveness however often suffer from overfitting. Recently convolutional neural networks have been proposed to allow expressive factorisation [references], do not suffer from overfitting and remain computationally efficient.

1.3.3 Other of Latent Feature Modelling Approaches

A number of alternative approaches to latent feature model factorisation have been proposed for link prediction, including circular correlation [reference], holographic entity-relational transformations [reference], toroidal representations [reference]. The rest of this dissertation focuses on factorisation of latent feature models, with the explicit representation of relational concepts.

Chapter 2

Literature Review

Statistical relational learning for knowledge graphs
Latent feature models
Link prediction

2.1 Knowledge Graphs

Knowledge graphs (KG) encode the relational information of a domain [reference]. KGs are composed of nodes and edges where nodes represent the entities and edges represent relations between entities. Two entities linked by a relation in a KG represents a fact, a KG is a collection of such facts represented as nodes and edges. Furthermore, the graphical structure of the representation models local, quasi-local and global properties about entities and relations. We can use direct entity relationships, or implicit structural properties to perform inference on the domain being modelled [reference]. For example
KGs are modelled under closed world or open world assumptions [reference]. Under a closed world assumption, the KG is considered complete, and the graph completely captures all facts within a domain. Under an open world assumption, it is possible to infer new facts about a domain from existing facts [reference]. It is also possible to infer new facts about a domain from the structural properties of the graph. Under an open world assumption, link prediction is used to generate new facts from existing ones. Graphs can thus be queried for existing facts, and can be 'reason' new facts. KGs have been applied to the field of information extraction [reference], search query augmentation [reference] and general question answering [reference].

Knowledge Bases
Ontologies

RDF

I'm going to randomly include a picture Figure 2.1.

If you have trouble viewing this document contact Krishna at: kks32@cam.ac.uk or raise an issue at https://github.com/kks32/phd-thesis-template/

Fig. 2.1 This is just a long figure caption for the minion in Despicable Me from Pixar

2.2 Matrix Factorization

Singular Value Decomposition Matrix factorisation is an approach used to compute latent features that capture relations between entities within a dataset [reference]. More formally, independent and identically distributed (IID) interactions between entities can be aggregated and represent in matrix format. The factorisation of the matrix will produce two real-valued unitary matrices that form a basis [reference], which can be thought of as a basis for a domain from where the data was sampled. Factorisation also produces a real-valued diagonal mastrix. The entries along the diagonal represent concepts captured by the dataset. A popular application of matrix factorisation is singular value decomposition [reference]. This is an implicit nodelling of relational domain concepts.

1. The first topic is dull
2. The second topic is duller
   (a) The first subtopic is silly
   (b) The second subtopic is stupid
3. The third topic is the dullest

Itemize

- The first topic is dull
- The second topic is duller
  - The first subtopic is silly

    - The second subtopic is stupid

- The third topic is the dullest

Description

The first topic is dull

The second topic is duller

> **The first subtopic** is silly
>
> **The second subtopic** is stupid

The third topic is the dullest

2.3 Tensor Factorisation

Tensor factorisation is an explicit modelling of entity relations using matrix slices of relational tensors [reference]. It involves modelling interactions of subject and object entities using relation-specific latent representations. These interactions have been modelled using the bilinlear tensor product, where the inner product of the subject entity is taken with a matrix relational representation, and the dot product of the generated embedding is then taken with the object entity. Bilinear tensor factorisation models are efficient in their number of parameters but lack expressiveness. Multilayer perceptrons were used to try overcome the lack of expressiveness however often suffer from overfitting. Recently convolutional neural networks have been proposed to allow expressive factorisation, do not suffer from overfitting and remain computationally efficient.
Tensor factorisation is a technique used in latent feature modelling to build representations of entities modified by their relationships. These entity-relational representations are then combined with other entities to score facts within the KG as well as the possibility of a new fact. Tensor factorisation models aim to be linearly scalable with datasets and so attempt to balance expressiveness with parameter efficiency.
RESCAL is a bilinear tensor model that combines latent entity-relational features of different entities using a matrix for each relation to compute a link score.
Distmult simplifies RESCAL by limiting the full rank relational matrix to a diagonal, with zeros everywhere else. Relational transformation of the entity is thus limited only to a stretch, limiting the expressiveness of the model whilst improving scalability.
TransE projects object entities e2 via relation-specific offsets and then aggregates the generated embedding features with the subject entity e1 and uses the new representation as input to a scoring function.

2.4 Nonlinear Factorisation

Parameterisation problem from tensor factorisation approach [nickel].
E-MLP ...
ER-MLP ...
Neural Tensor Networks extend the bilinear tensor product by concatenating the two entities and applying a weight operation on the concatenation to generate a latent representation which is then applied to a fully-connected neural layer to generation a score. HolE uses holographic embeddings described as circular correlations to compute triple scores. This

involves taking the circular product of the interactions between each entity, and then taking the product of the generated representation along with a transposed relational vector.
ComplEx ...
HolE ...
ConvE provides a more expressive model by taking the convolution of triple entities with a 2-dimensional convolutional relational model. This allows more expressive latent representations to be computed, whilst using a CNN architecture for parameter tying and parameter efficiency.
HypER simplifies ConvE by using 1-dimensional convolutional relational filters. The convolution of the relational filter is then taken with the subject entity e1, before a dot product with the object entity e2 generates a triple score.

(a) Tom and Jerry (b) Wall-E (c) Minions

Fig. 2.2 Best Animations

Subplots

I can cite Wall-E (see Fig. 2.2b) and Minions in despicable me (Fig. 2.2c) or I can cite the whole figure as Fig. 2.2

Chapter 3

Link Prediction With Latent Feature Modelling

3.1 Link Prediction

And now I begin my third chapter here . . . And now to cite some more people **? ?**]

Link prediction aims to rank plausible entity relationships. In latent feature modelling, it is posed as a classification task. These ranks are computed by scoring entity-relational pairs against all entities within the knowledge base. Relationships that rank highly become potential new facts that have been discovered and generate new knowledge within the knowledge base. Similar to inductive logic **? ?**], these facts need to be coherent, consistent and are not contradictory. Latent feature models contain statistical properties such as the capability to model a likelihood distribution over candidate entity relationships. This is.a more flexible modelling approach that provides a measure "confidence" in a fact, given the magnitude of the computed relationship score.

3.1.1 Entity-Relational Classification

. . . and some more

Entity-relational classification is a subsection of the more general ranking problem in machine learning (ML) **? ?**]. Standard classification involves the determining of the most appropriate categories in which entity belongs. If a logistic approach is used, a logit distribution across candidate classes is generated. A logit is referred to as an inverse probability **?**], a magnitude with is the passed thought a thresholding logarithmic sigmoid function to generate a likelihood, or a probability in a finite number of mutually exclusive classes..

The category with the highest logit associated with is then determined to be the most likely category in which the entity belongs. In the ranking.

3.1.2 Deep Learning Classification Models

... and some more ... The state-of-the-art in deep learning classification models is convolutional neural networks (CNN) **?**]. CNNs are typically applied to object classification within images, and take advantage of the properties of object translation invariance and object location invariance [**?**]. These properties allow CNNs to perform robust classifications that generalise well across datasets. Translation invariance allows the CNN to build an object representation that is consistent under transformation, for example an image rotation would lead to a consistent final object representation being generated prior to logit computation for classification. Locality invariance allows the model to correctly identify an object no matter where it may reside with the boundaries of an image.
These properties are possible because of how the final object representation is generated. CNNs perform a convolutional operation on an image, using a trainable image filter [**?**]. The operation generates feature map of the image that constructs latent object representations of the image. In practise these representations are chained together to produce more complex latent representations the deeper the CNN.
It is possible to decompose convolution operations into spatial and depth-wise convolutions [**?**]. A spatial convolution operates on different regions of an image, producing distinct representations for each region, for example and image can be divided into four regions, where a spatial convolution operates independently on each of these regions using region-specific filters.This operation produces four distinct feature maps which are later flattened into a single hidden layer representation, before begin run through a linear layer to generate the final logits. In a depth-wise convolution, the distinct convolutional feature maps are generated using the depth dimensions of the input image, typically three dimensions with images, the red, blue and green image channels in a colour image. Channel-specific filters are used to generate these feature maps and once again the future maps are flattened prior to computing logits for the model classes.

First subsub section in the second subsection

... and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it ...

3.1.3 Classification Model Development Framework

. . . and some more . . .

First subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more . . .

Second subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

3.2 Nonlinear Factorisation Models

and here I write more . . .

3.2.1 Models

. . . and some more what they are, and how we got here, trial and error

3.2.2 Training Algorithm

. . . and some more . . .

First subsub section in the second subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

3.2.3 Model Analysis

. . . and some more . . .

First subsub section in the third subsection

…and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more …

Second subsub section in the third subsection

…and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it …

3.3 Deep Learning Best Practise

This section has been modified from "Publication quality tables in LaTeX*" by Simon Fear.

The layout of a table has been established over centuries of experience and should only be altered in extraordinary circumstances.

When formatting a table, remember two simple guidelines at all times:

1. Never, ever use vertical rules (lines).

2. Never use double rules.

These guidelines may seem extreme but I have never found a good argument in favour of breaking them. For example, if you feel that the information in the left half of a table is so different from that on the right that it needs to be separated by a vertical line, then you should use two tables instead. Not everyone follows the second guideline:

There are three further guidelines worth mentioning here as they are generally not known outside the circle of professional typesetters and subeditors:

3. Put the units in the column heading (not in the body of the table).

4. Always precede a decimal point by a digit; thus 0.1 *not* just .1.

5. Do not use 'ditto' signs or any other such convention to repeat a previous value. In many circumstances a blank will serve just as well. If it won't, then repeat the value.

A frequently seen mistake is to use '\begin{center}' … '\end{center}' inside a figure or table environment. This center environment can cause additional vertical space. If you want to avoid that just use '\centering'

Table 3.1 A badly formatted table

	Species I		Species II	
Dental measurement	mean	SD	mean	SD
I1MD	6.23	0.91	5.2	0.7
I1LL	7.48	0.56	8.7	0.71
I2MD	3.99	0.63	4.22	0.54
I2LL	6.81	0.02	6.66	0.01
CMD	13.47	0.09	10.55	0.05
CBL	11.88	0.05	13.11	0.04

Table 3.2 A nice looking table

Dental measurement	Species I		Species II	
	mean	SD	mean	SD
I1MD	6.23	0.91	5.2	0.7
I1LL	7.48	0.56	8.7	0.71
I2MD	3.99	0.63	4.22	0.54
I2LL	6.81	0.02	6.66	0.01
CMD	13.47	0.09	10.55	0.05
CBL	11.88	0.05	13.11	0.04

Table 3.3 Even better looking table using booktabs

Dental measurement	Species I		Species II	
	mean	SD	mean	SD
I1MD	6.23	0.91	5.2	0.7
I1LL	7.48	0.56	8.7	0.71
I2MD	3.99	0.63	4.22	0.54
I2LL	6.81	0.02	6.66	0.01
CMD	13.47	0.09	10.55	0.05
CBL	11.88	0.05	13.11	0.04

3.4 Loss Surface Analysis

and here I write more . . .

3.4.1 Models

. . . and some more what they are, and how we got here, trial and error

3.4.2 Training Algorithm

. . . and some more . . .

First subsub section in the second subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

3.4.3 Model Analysis

. . . and some more . . .

First subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more . . .

Second subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

3.5 Nonlinear Factorisation Hypotheses

and here I write more . . .

3.5.1 Models

. . . and some more what they are, and how we got here, trial and error

3.5.2 Training Algorithm

…and some more …

First subsub section in the second subsection

…and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it …

3.5.3 Model Analysis

…and some more …

First subsub section in the third subsection

…and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more …

Second subsub section in the third subsection

…and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it …

Appendix A

How to install LaTeX

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from
   https://www.tug.org/texlive/

2. Download WinCDEmu (if you don't have a virtual drive) from
   http://wincdemu.sysprogs.org/download/

3. To install Windows CD Emulator follow the instructions at
   http://wincdemu.sysprogs.org/tutorials/install/

4. Right click the iso and mount it using the WinCDEmu as shown in
   http://wincdemu.sysprogs.org/tutorials/mount/

5. Open your virtual drive and run setup.pl

or

Basic MikTeX - TeX distribution

1. Download Basic-MiKTeX(32bit or 64bit) from
   http://miktex.org/download

2. Run the installer

3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin) and choose Package Manager

4. Select or search for packages to install

TexStudio - TeX editor

1. Download TexStudio from
http://texstudio.sourceforge.net/#downloads

2. Run the installer

Mac OS X

MacTeX - TeX distribution

1. Download the file from
https://www.tug.org/mactex/

2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - TeX editor

1. Download TexStudio from
http://texstudio.sourceforge.net/#downloads

2. Extract and Start

Unix/Linux

TeXLive - TeX distribution

Getting the distribution:

1. TexLive can be downloaded from
http://www.tug.org/texlive/acquire-netinstall.html.

2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

   ```
   mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
   ```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

   ```
   cd /your/download/directory
   ./install-tl
   ```

4. Enter command 'i' for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64bit OS

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
```

```
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CentOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.