A UTILITY-DRIVEN APPROACH TO INSTANCE-BASED TRANSFER LEARNING FOR BOOSTED RELATIONAL DEPENDENCY NETWORKS

Cainã Figueiredo Pereira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Gerson Zaverucha
Daniel Sadoc Menasché

Rio de Janeiro
Agosto de 2024

A UTILITY-DRIVEN APPROACH TO INSTANCE-BASED TRANSFER LEARNING FOR BOOSTED RELATIONAL DEPENDENCY NETWORKS

Cainã Figueiredo Pereira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Gerson Zaverucha
Daniel Sadoc Menasché

Aprovada por: Prof. Gerson Zaverucha
Prof. Daniel Sadoc Menasché
Prof. Aline Marins Paes Carvalho
Prof. Valmir Carneiro Barbosa

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2024

Figueiredo Pereira, Cainã

A Utility-Driven Approach to Instance-Based Transfer Learning for Boosted Relational Dependency Networks/Cainã Figueiredo Pereira. – Rio de Janeiro: UFRJ/COPPE, 2024.

XI, 71 p.: il.; 29, 7cm.

Orientadores: Gerson Zaverucha

Daniel Sadoc Menasché

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2024.

Referências Bibliográficas: p. 54 – 59.

1. Instance-Based Transfer Learning. 2. Statistical Relational Learning. 3. Utility Theory. I. Zaverucha, Gerson *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho à minha
família, por estar ao meu lado
em cada passo desta jornada.

Agradecimentos

Gostaria de começar expressando minha profunda gratidão a Deus, pela oportunidade de chegar até aqui e por ter me dado força para concluir mais uma importante etapa da minha vida.

Agradeço aos meus orientadores, Gerson Zaverucha e Daniel Sadoc, pelo apoio contínuo e parceria que vem desde a graduação, pelo incentivo para ingressar no mestrado e pela orientação indispensável ao longo de todo período. O aprendizado proporcionado por vocês foi fundamental para o desenvolvimento desta dissertação.

Sou imensamente grato à professora Aline Paes e ao professor Valmir Barbosa, que, desde as etapas iniciais da dissertação, se dispuseram a participar ativamente deste projeto, contribuindo com novas ideias e sugestões valiosas, que foram essenciais para refinar meu trabalho.

Agradeço também ao professor Valmir Barbosa pelo acesso às máquinas em seu laboratório, e aos responsáveis pelo LAND pelo acesso ao seu cluster. Em especial, sou grato ao Davi Brilhante, que sempre me ofereceu suporte com o uso do cluster.

À Thais e ao Rodrigo, meu sincero agradecimento por estarem sempre dispostos a ajudar, esclarecendo dúvidas sobre seus trabalhos e oferecendo sugestões para o desenvolvimento do meu.

Agradeço à toda a equipe do Programa de Engenharia de Sistemas e Computação (PESC), que desempenhou um papel crucial nesta etapa da minha formação.

Finalmente, minha eterna gratidão à minha família, que esteve ao meu lado em todos os momentos, oferecendo o incentivo e o apoio necessários para a realização deste trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

A UTILITY-DRIVEN APPROACH TO INSTANCE-BASED TRANSFER LEARNING FOR BOOSTED RELATIONAL DEPENDENCY NETWORKS

Cainã Figueiredo Pereira

Agosto/2024

Orientadores: Gerson Zaverucha
Daniel Sadoc Menasché

Programa: Engenharia de Sistemas e Computação

Aprendizado Estatístico Relacional explora um espaço de busca complexo de objetos, relações e parâmetros probabilísticos para otimizar um modelo. Para reduzir a complexidade da busca e os requisitos de disponibilidade de dados, trabalhos relacionados exploraram a transferência de um modelo aprendido em um domínio de origem para um domínio de destino. No entanto, esses modelos nem sempre estão disponíveis, e um aprendizado imperfeito no domínio de origem pode prejudicar o desempenho no domínio de destino. Este trabalho propõe utilizar as instâncias de um domínio de origem ao invés de um modelo aprendido nesse domínio. Uma solução simples, como concatenar instâncias de ambos os domínios, pode ser ineficaz devido ao impacto potencial de instâncias irrelevantes ou de baixa qualidade. Para abordar essa questão, nós tratamos a seleção de instâncias como sendo uma tarefa de alocação justa de recursos, onde as utilidades associadas às instâncias são parametrizadas para capturar a relevância de cada instância. Nós introduzimos o método chamado UTIL-BRDN, que aplica essa abordagem baseada em utilidades às Redes de Dependência Relacional com *Boosting* (RDN-Boost). Nossos resultados experimentais mostram que o UTIL-BRDN transfere conhecimento de forma eficaz reutilizando instâncias de outros domínios e é robusto contra *negative transfer*. Nossas contribuições incluem a introdução do aprendizado por transferência baseado em instâncias no contexto do aprendizado estatístico relacional, o desenvolvimento de uma abordagem baseada em utilidades para a seleção de instâncias, a extensão do RDN-Boost para lidar com múltiplos domínios e utilidades, e a realização de uma extensa avaliação empírica do método proposto.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A UTILITY-DRIVEN APPROACH TO INSTANCE-BASED TRANSFER LEARNING FOR BOOSTED RELATIONAL DEPENDENCY NETWORKS

Cainã Figueiredo Pereira

August/2024

Advisors: Gerson Zaverucha
Daniel Sadoc Menasché

Department: Systems Engineering and Computer Science

Statistical Relational Learning explores a complex search space of objects, their relationships, and probability parameters to find an optimal model. To reduce search complexity and data availability requirements, previous work has explored taking advantage of a learned model in a source domain and *transferring* it to a target domain. However, these models are not always available and imperfect learning in the source domain can hinder the performance in the target domain. This work proposes to leverage the instances of a source domain instead of its learned model. A simple solution, such as concatenating instances from both domains, is likely ineffective due to the potential negative impact of irrelevant or poor-quality instances. We address this by framing instance selection as a task of fair resource allocation, where utilities associated with instances are parameterized to capture the relevance of each instance. We introduce a method called UTIL-BRDN, which applies this utility-driven approach to Boosted Relational Dependency Networks (RDN-Boost). Our experimental results show that UTIL-BRDN effectively transfers knowledge by reusing instances from other domains and is robust against negative transfer. Our contributions include introducing instance-based transfer learning to statistical relational learning, developing a utility-driven approach to instance selection, extending RDN-Boost to handle multiple domains and utilities, and conducting an extensive empirical evaluation of the proposed method.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Statistical Relational Learning (SRL) is a machine learning subarea that considers the relationships between objects when building statistical models to solve a given task [1, 2]. However, to effectively incorporate those relationships into the model, SRL must traverse a complex search space of possible connections and interactions between the objects [3]. Finding a model in those circumstances demands large volumes of data describing the object's properties and possible relations among them. On the other hand, relational data from different domains can share common patterns or features, which is a key principle in transfer learning [4]. Transfer learning leverages this shared information to transfer knowledge learned from one domain to another, enabling models to generalize more effectively across different tasks or contexts, even when training data are limited in the target domain.

1.1 Prior Art

Previous works have relied on model-based transfer learning to induce SRL models using limited data [5–7]. Although these approaches utilize different SRL languages as the backbone, they share the common assumption that a learned source model can be adapted to fit the target domain. However, because the source model is optimized for the source domain, it may not effectively capture features that are relevant to both the source and target domains. Moreover, the adaptation process is often computationally expensive [6].

In contrast, instance-based transfer learning [4, 8, 9] focuses on reusing or reweighting instances from the source domain for the target task, allowing the target model to adapt directly from actual examples. A key advantage of this approach is that it does not depend on a pre-trained model but instead adjusts the relevance of individual instances. However, a critical challenge arises: *How can we numerically define the importance of source and target instances when learning the target model?* In this work, we address this challenge by designing an instance selection mechanism

inspired by the framework of utility maximization, which is typically used to solve problems of fair resource allocation [10–14].

Fairness is a key dimension in the learning process and has been incorporated into classical problems such as fair bandwidth allocation in computer networks and, more generally, fair resource allocation in computer systems. Different fairness definitions are captured by utility functions. A notable utility function in this context is the α-*fairness* [15, 16]. To the best of our knowledge, no previous work has applied a utility-based framework to instance-based transfer learning in relational domains.

1.2 Goals and Research Questions

This work aims to leverage instance-based transfer learning and utilities to effectively transfer knowledge across relational domains to improve the performance of a target SRL model. To this end, we propose a two-stage transfer approach called UTIL-BRDN. UTIL-BRDN first maps and combines data from two distinct relational domains and then uses the combined data to fit a SRL model, augmented with utilities, for the target task. In particular, we focus on Boosted Relational Dependency Networks (RDN-Boost), an alternative strategy to learning Relational Dependency Networks (RDNs).

Throughout this work, we aim to answer the following research questions regarding the properties of UTIL-BRDN and its relation to the baseline consisting of learning from scratch:

- **Q1.** Does UTIL-BRDN learn accurate models that outperform learning from scratch, by transferring instances from another domain?
- **Q2.** Does the proposed utility-based objective function impact transfer learning?
- **Q3.** Does UTIL-BRDN learn accurate models despite poor mapping choices?
- **Q4.** Which aspects of a mapping contribute to successful transfer to a target domain?
- **Q5.** Which mechanisms does UTIL-BRDN have to mitigate negative transfer?
- **Q6.** Does balancing across instance groups effectively enhance performance?
- **Q7.** How sensitive is UTIL-BRDN to the amount of data available for the target domain?

This work does not address the problems of best mapping search and source selection.

1.3 Key Results

We conducted several experiments to address our research questions. First, we established baselines by evaluating RDN-Boost in learning from scratch on seven widely used real-world relational datasets, considering scenarios with both large and limited data availability. We then performed three different sets of transfer experiments to evaluate the properties of UTIL-BRDN, aiming to understand how the model's performance relates to the use of utilities, the selected mapping, the instance weighting strategy, and the amount of data available in the target domain.

Our results indicate that UTIL-BRDN effectively transfers knowledge by leveraging instances from the source domain (**Q1**). With appropriate parameterization, found via grid search, UTIL-BRDN significantly outperforms the baseline in certain transfer scenarios, such as from IMDB to Cora, with an improvement of 0.31 in AUC-PR, and from Twitter to Yeast, whose improvement in AUC-PR is 0.18. Interestingly, Cora and Yeast are the domains most affected by data scarcity in learning from scratch and also benefit the most from transfer with UTIL-BRDN. In other scenarios, our method achieves results comparable to learning from scratch. We demonstrate that this capability is closely related to the use of utilities (**Q2**). In particular, the UTIL-BRDN variant without these utilities incorporated into RDN-Boost is less robust and efficient.

We also find evidence indicating that UTIL-BRDN presents some robustness to poor mapping choices (**Q3**). We visually illustrate this property, showing that certain mappings consistently yield models superior to the baseline. In contrast, other mappings are much more dependent on the model parameterization and often result in worse models. Despite that, under an appropriate parameterization of UTIL-BRDN, even bad mappings can lead to models as good as those obtained with high-quality mapping. We also provide empirical evidence on the relation between source-target similarity, which depends on the mapping, and UTIL-BRDN performance (**Q4**).

Moreover, we found that utilities and instance weights provide UTIL-BRDN with both implicit and explicit mechanisms for instance selection, respectively. This is instrumental in mitigating the negative transfer and enhancing the performance [8] (**Q5** and **Q6**). On the other hand, the amount of target data mainly impacts UTIL-BRDN models with poor parameterization but is not enough to significantly improve learning from scratch (**Q7**).

1.4 Contributions

In summary, our main contribution is threefold.

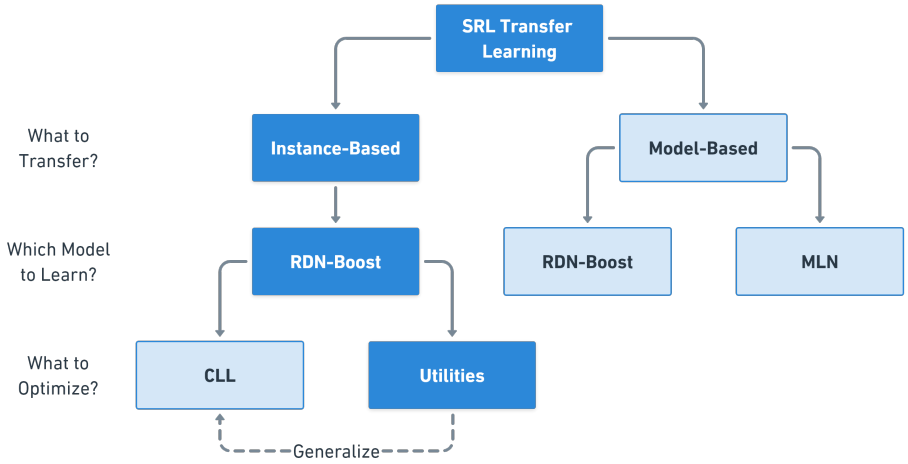


Figure 1.1: SRL transfer learning taxonomy.

1. **Instance-based transfer learning for the relational domain:** We introduce the approach known as instance-based transfer learning into relational domains, focusing on statistical relational models. In particular, our approach leverages the data from a source domain instead of leveraging a pre-trained model.

2. **Utility-based learning for the relational domain:** This work focuses on RDN-Boost and extends it to accommodate a broader class of utility functions beyond its traditional objective function consisting of the conditional log-likelihood (CLL). We achieve this by introducing α-fair utilities, which have max-throughput fairness, proportional fairness, and max-min fairness as special cases. In particular, CLL corresponds to proportional fairness under our design. Utilities can vary across domains and capture non-linear weights, reflecting that the cost of errors can be tuned in an instance-based fashion.

3. **Assessment of the impact of utilities on classification accuracy and robustness:** Given a target metric of interest, such as AUC-ROC or AUC-PR, we report experimental results indicating how utilities impact such metrics over a broad set of domains. In particular, we experimentally indicate that utility functions can compensate for bad mappings between source and target domains. Indeed, utilities build robustness into the learning process, providing a flexible mechanism to mitigate negative transfer.

Figure 1.1 identifies our key contributions (dark blue) among related literature approaches for SRL transfer learning.

1.5 Outline

The remainder of this work is organized as follows. Chapter 2 reviews the literature on statistical relational learning and other relational learning approaches, transfer learning, and utility maximization. Next, we introduce some basic background on these fields in Chapter 3. We also introduce Relational Dependency Networks (RDNs) and its learning algorithm based on functional gradient boosting (RDN-Boost). Chapter 4 describes UTIL-BRDN, our utility-driven approach to instance-based transfer learning for statistical relational learning. The experiments and their results are presented and discussed in Chapter 5. Finally, Chapter 6 concludes this work and suggests possible directions for future work.

Chapter 2

Related Work

2.1 Relational Learning

For a long time, much of the focus in machine learning was on propositional techniques. However, due to the complexity of relational data, propositional models are unable to effectively learn from this type of data. To address this limitation, several areas of machine learning have developed strategies to enable learning from relational data. Figure 2.1 provides an overview of key contributions in this field, including Propositionalization, Knowledge Graph Embeddings, Graph Neural Networks, Inductive Logic Programming, and Statistical Relational Learning.

Propositionalization. Propositionalization [17–19] involves pre-processing relational data to reduce it into a propositional representation, typically starting with multiple tables (relational databases) or First-Order Logic (FOL) knowledge bases. In the propositional format, instances are characterized by a fixed-length vector of attribute-value pairs, which forms a single table that can be used to train traditional machine learning models, such as neural networks and decision trees. The ability to leverage traditional models makes propositionalization an attractive alternative for learning in relational domains. However, the conversion of data representation typically results in a loss of information, which can be detrimental to the performance of the resulting model.

Knowledge Graph Embeddings and Graph Neural Networks. Rather than framing relational data into a propositional representation to fit traditional models, some approaches extend propositional models to the relational domain. Knowledge Graph Embeddings (KGE) and Graph Neural Networks (GNNs) both process graphs as input and learn vector representations (embeddings) for nodes and edges.

In KGE [20–22], the learned representations allow inferences to be made through algebraic operations on the embeddings, such as vector addition and projection.

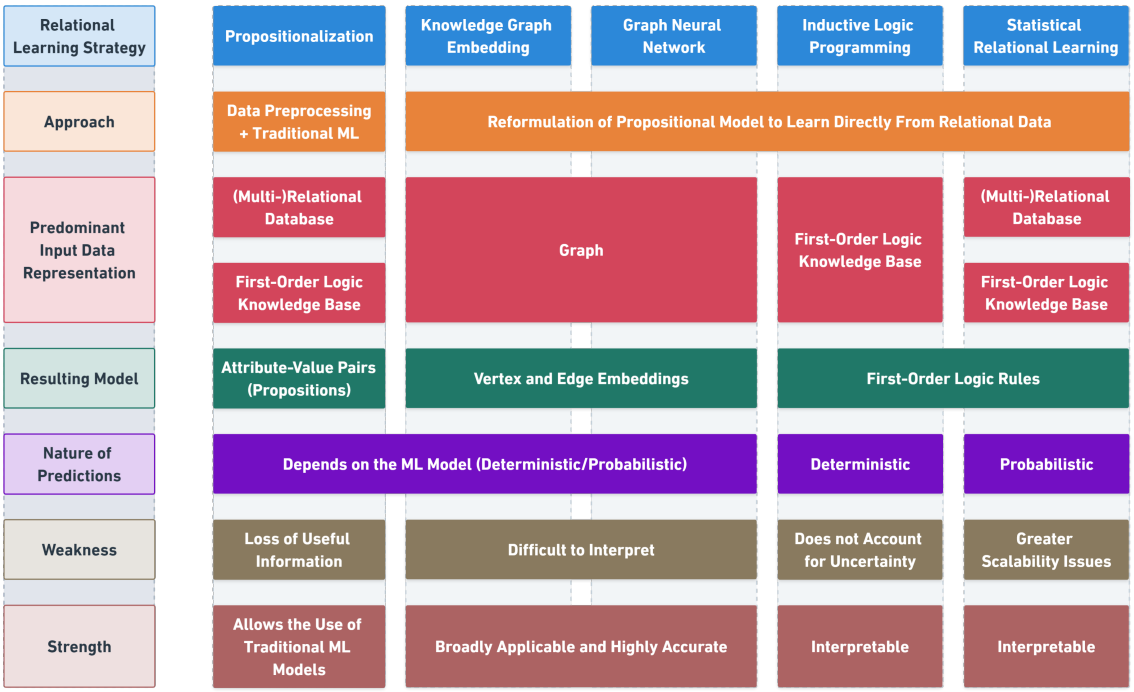


Figure 2.1: Comparison of Relational Learning Strategies

These methods are commonly used in tasks like knowledge graph completion, where missing relationships are inferred based on the learned patterns in the embeddings.

On the other hand, GNNs [23] are a category of deep neural networks designed to operate on graph structures using special convolution operators based on the message-passing framework. Stacking GNN layers, such as Graph Convolutional Network (GCN) [24] and Graph Attention Network (GAT) [25], allows for the extraction of expressive embeddings for the graph's nodes and edges. GNNs have proven effective in tasks such as node classification, link prediction, graph-level prediction, community detection, and anomaly detection. While these methods typically achieve high accuracy, one limitation is their lack of interpretability compared to logic-based approaches.

Inductive Logic Programming. Inductive Logic Programming (ILP) [26–29] extends First-Order Logic (FOL) to enable the learning of rules from relational data, typically represented in a Prolog-like format. Prominent ILP models include FOIL (First-Order Inductive Learner) [27], Progol [28], and TILDE [29], which extends decision trees to relational domains. One of the key strengths of ILP methods is their interpretability. However, ILP does not account for uncertainty, which can be a limitation in many real-world applications.

Statistical Relational Learning. To overcome the limitations of ILP in handling uncertainty, SRL [1, 2] incorporates probabilistic reasoning into relational models.

SRL methods usually extend graphical models to relational domains, learning probabilistic distributions over attributes of different objects in the domain, as specified in the framework of Probabilistic Relational Models [30]. Examples of SRL models include Relational Bayesian Networks [30] and Relational Dependency Networks [31], which assume relational data is represented across multiple tables in a relational database. Other models, such as Markov Logic Networks [32] and Boosted Relational Dependency Networks [33], are designed to operate over First-Order Logic knowledge bases. SRL models offer high expressiveness and can typically be interpreted, but they often face scalability challenges due to the large search space, the complexity of model structure learning, and the inference procedures involved.

This work focuses on the field of Statistical Relational Learning, particularly on Boosted Relational Dependency Networks, as it provides an efficient alternative for learning both the structure and parameters of Relational Dependency Networks [33] and achieved superior results compared to other state-of-the-art approaches [33].

2.2 Instance-Based Transfer Learning

Instance-based transfer learning is an alternative approach to performing transfer learning [8, 9, 34]. In this approach, instances from the source domain are integrated with the dataset from the target domain, and a model is fitted using the combined data.

TrAdaBoost [9] is an instance-based transfer learning extension of AdaBoost. Like AdaBoost, it dynamically adapts the weights of instances based on model predictions. However, TrAdaBoost introduces a mechanism to differentiate between instances from different domains: misclassified instances in the source domain are punished, while those in the target domain have their weights increased. This approach aims to prevent the model from learning spurious patterns from the source domain while encouraging it to improve predictions for misclassified instances in the target domain.

Our work shares similarities with TrAdaBoost [9] in at least two aspects. First, both methods fall within the category of instance-based transfer learning, transferring instances from the source to the target domain. Second, our method also includes hyperparameters that determine how to handle poor instances from each domain, particularly those with large prediction errors. Consequently, our approach can similarly punish misclassified instances from the source domain or emphasize those from the target domain. Other strategies are also feasible under our approach.

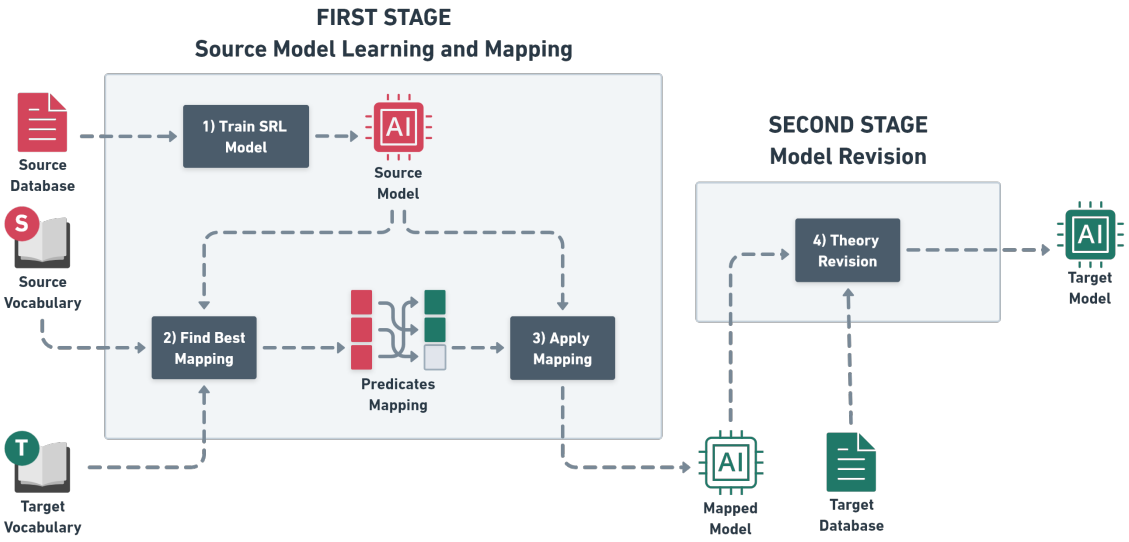


Figure 2.2: Typical transfer learning pipeline in SRL: learn a model from source data, map it to the target domain, and refine with target data.

2.3 Transfer in Statistical Relational Learning

Figure 2.2 shows the pipeline usually employed in the literature of transfer learning for statistical relational learning. It consists of the following separate and independent stages: model learning from source domain, model mapping, and revision. In the first stage, the structure of a model trained on the source domain data are mapped to the target domain. Next, this mapped model is refined using a theory revision method using the target domain data. The strategy of reusing a model pre-trained in another domain to learn a task in the target domain is also known as model-based transfer learning. Works following this pipeline vary primarily in their choice of the predictive model for the transfer process (step 1, in Figure 2.2), the approach used to map the model structure (steps 2 and 3, in Figure 2.2), and the techniques employed for the model refinement (step 3, in Figure 2.2). Most of the existing approaches are tailored to work with Markov Logic Networks (MLNs) [5, 32, 35–38] or RDN-Boost [6, 7, 39, 40].

Transfer of Markov Logic Networks. To our knowledge, TAMAR (Transfer via Automatic Mapping And Revision) [5, 32] is the pioneering approach in transfer learning for SRL models. Using a best-mapping heuristic, TAMAR autonomously maps the clauses of a MLN learned from the source domain to a target domain. For each clause, it exhaustively searches all valid predicate mappings and selects the one that maximizes the weighted pseudo-log-likelihood (WPLL) on the target data. Following this, it refines the mapped model on the target domain using an algorithm similar to FORTE [41].

SR2LR (Short-Range to Long-Range) [35] adapts TAMAR for scenarios with

extremely limited target data, where all available information pertains to a single entity. SR2LR omits the model revision step and instead focuses on mapping predicates from clauses directly applicable to the single target entity (short-range clauses) before addressing the remaining clauses (long-range clauses). This method has been shown to significantly improve performance compared to mTAMAR, which handles only the mapping phase of TAMAR.

Other transfer approaches involve learning domain-independent patterns from the source domain and encoding them as second-order Markov logic formulas, which are then instantiated with target domain predicates [36, 37]. Unlike first-order logic formulas specific to a domain, second-order formulas include predicate variables that can be instantiated across domains. By using this approach, DTM (Deep Transfer via Markov logic) DAVIS e DOMINGOS [36] addresses TAMAR's scalability issues by avoiding exhaustive searches during clause mapping.

TODTLER (Two-Order-Deep Transfer Learning) [37] builds on the concepts of learning and transferring second-order formulas, outperforming DTM in both accuracy and efficiency. TODTLER employs a three-level dataset sampling process: (1) second-order model ($M^{(2)}$) for selecting second-order templates, (2) first-order model ($M^{(1)}$) for instantiating second-order formulas into first-order ones, and (3) dataset (D) for sampling ground facts from $M^{(1)}$. The transfer strategy updates the prior of the second-order model with its posterior from the source domain, thereby biasing the learning of the first-order model in the target domain.

In [38], transfer learning is also explicitly conceptualized as the introduction of bias. Their Language-bias Transfer Learning (LTL) method constructs a matching-type tree from clauses learned in the source domain, using this tree to constrain the search space in the target domain. The search tree is then converted into clauses, which are refined through a theory revision step to improve performance in the target domain. This approach ensures that the transferred knowledge effectively guides the learning process in the target domain.

Transfer of Boosted Relational Dependency Networks. The use of Relational Dependency Networks for SRL transfer learning was first introduced by the method TreeBoostler [6]. This method adapts the mapping and revision processes to account for the specific structure learned by RDN-Boost, consisting of a series of Relational Regression Trees (RRTs). Unlike MLNs, where predicate mappings are independent across clauses, TreeBoostler maps predicates within each tree node according to their order in the tree and adjusts subsequent nodes based on the already mapped predicates. The revision process involves pruning or expanding nodes to enhance the model.

TransBoostler [7] offers an alternative mapping strategy to TreeBoostler by lever-

aging word embeddings. This approach uses pre-trained word embeddings to extract vector representations for predicates and performs mapping based on vector similarity. This method not only achieves strong performance but also tends to be more time-efficient by utilizing the contextual information encoded in the embeddings.

GROOT (Genetic algorithms to aid tRansfer learning with bOOsTsrl) [39] extends TreeBoostler by incorporating genetic algorithms for transfer learning. Unlike local search methods, GROOT uses genetic algorithms to transfer the source model structure to the target domain while simultaneously refining the model. Each individual in the genetic algorithm's population represents a mapping of trees, with operators such as mutation (pruning or expanding branches) and crossover (exchanging subtrees between individuals) driving the evolution of the model.

Despite all these works focusing on transfer learning in SRL, little attention has been given to investigating how to determine the utility of a source domain for a given target domain before even performing the transfer. The work of [40] aims to fill this gap, indicating how to choose the source domain in case one has the flexibility to do so.

Our work: Instance-Based Transfer Learning for Relational Domains. Similar to the works discussed in this section, we also address the problem of transfer learning in relational domains using SRL models. In particular, we follow recent works on the literature [6, 7, 39, 40] and build our approach upon RDN-Boost.

However, we introduce a novel framework that transfers instances from the source domain to the target domain, rather than transferring the structure of a model learned from the source domain. In the relational realm, this strategy, known as instance-based transfer learning, still relies on mapping but eliminates the need for a revision step, as the target model is simultaneously learned from both target and source instances. This is possible because we design a utility-based objective function for the RDN-Boost algorithm that integrates instances from both domains, allowing us to control the influence of each instance on the learning process. Unlike some of the previous work, we do not propose or explore different mechanisms to find the best mapping. Instead, we assess the properties of our approach.

Moreover, we conduct an investigation complementary to the work of LUCA *et al.* [40], examining how the similarity between source and target data impacts model performance in a transfer learning setting. Unlike their approach, which proposes a new algorithm to select the best source domain, we evaluate the robustness of our transfer method in the presence of source data with very disparate distributions. Additionally, we introduce the interpretation of mappings as different source data.

Chapter 3

Background

This chapter introduces some of the key concepts used in this work. Section 3.1 introduces the basic concepts of First-Order Logic and characterizes a knowledge base. Next, Section 3.2 provides an overview of transfer learning, specifically focusing on instance-based transfer learning. Then, Section 3.3 presents the ILP model called Relational Regression Tree (RRT). In Section 3.4, we show how RRTs are used by RDN-Boost, a gradient-based boosting approach used to learn RDNs. Finally, Section 3.5 describes the utility maximization framework.

3.1 First-Order Logic

First-Order Logic is a powerful extension of propositional logic, well-suited for capturing the complexity of relational domains and enabling reasoning. Objects from the domain, such as the object *mary*, are represented by *constants*, which are denoted by lowercase letters. A *variable*, such as *X*, is represented by an uppercase letter and corresponds to a generic object that can be replaced by a constant. Both constants and variables are referred to as *terms*. Each term is associated with a *type*.

A *predicate* defines a relationship between terms. It is denoted by the predicate name in lowercase letters, followed by its *arity* (e.g., *parent_ of/2*). The arity specifies the number of terms the predicate takes as arguments. For instance, in *parent_ of/2*, the arity is 2, meaning the predicate expects two terms: the parent and the child. Predicates of arity 1 and 2 are called *unary* and *binary* predicates, respectively. Unary predicates, such as *actor/1*, typically assign a label to an object, while binary predicates can encode either the attributes of that object (e.g., *age/2*) or relationships between different objects (e.g., *parent_ of/2*).

An *atom* is a predicate with explicit terms, such as in *parent_ of(sheila, X)*. An atom is also called a *positive literal*. When preceded by a negation operator (e.g., ¬*parent_ of(sheila, X)*), it is referred to as a *negative literal*. Literals (or

atoms) without variables (e.g., *parent_ of(cynthia, mary)*) are called *ground literals* (or *ground atoms*).

A knowledge base for a predictive task is built of a set of labeled instances and a background knowledge about the domain. The labeled instances include positive and negative examples of the target relation. In turn, the background knowledge records logical facts characterizing the objects and relationships within the domain. Additionally, a set of mode specifications is typically required by First-Order Logic-based learning algorithms to efficiently explore the model search space [28, 29]. A mode specifies the types of terms for each predicate and how the variables can be bound in the rules the model explores during learning.

Table 3.1 provides an example of such a base, where the task is to predict whether one person is the sister of another (*sister_ of/2*), using information about their genders (*male/1* or *female/1*), parents (*parent/2*), and workplaces (*works_ at/2*).

Table 3.1: A toy example of a First-Order Logic knowledge base

Positives	**Negatives**	**Facts**	**Modes**
sister_of(sheila, eva).	sister_of(sheila, peter).	female(sheila).	female(+person).
sister_of(sheila, david).	sister_of(david, jessica).	female(eva).	male(+person).
sister_of(sheila, ron).	sister_of(zula, amanda).	female(cynthia).	parent_of(+person, +person).
sister_of(cynthia, steven).	sister_of(donald, cynthia).	male(ron).	parent_of(-person, +person).
sister_of(mary, mark).	sister_of(jessica, ron).	parent_of(jessica, sheila).	parent_of(+person, -person).
sister_of(zula, clevend).	sister_of(mark, cynthia).	parent_of(jessica, eva).	sister_of(+person, +person).
...	...	parent_of(cynthia, mark).	sister_of(-person, +person).
		parent_of(cynthia, mary).	sister_of(+person, -person).
		works_at(ron, google)	works_at(+person, +company).
		works_at(jessica, microsoft)	works_at(-person, +company).
		...	works_at(+person, -company).

3.2 Transfer Learning

Transfer learning [42–45] focuses on transferring knowledge gained from a task on a *source* domain to improve the performance of another task, often related to the first one, on a *target* domain. It is particularly useful when the target domain lacks sufficient data to effectively train an accurate model using traditional machine learning methods, and augmenting the dataset is either expensive, time-consuming, or infeasible. In this work, we refer to traditional learning methods as *learning from scratch*, where a new model is trained independently for each task without leveraging knowledge from other tasks.

Beyond the assumption that the number of samples in the target dataset is much smaller than in the source dataset, other common assumptions are summarized in the following categorization of transfer learning methods:

- **Homogeneous transfer learning.** this category applies when the source and target domains share some common features and their tasks have the same set

of possible labels. Although the tasks or domains are related, they still differ in terms of how the data is distributed or how the labels are associated with the features. For example, the same features may behave differently in the two domains, or the relationship between the features and the labels might change.

- **Heterogeneous transfer learning.** This occurs when the source and target domains do not share any common features or their tasks have different sets of possible labels.

Transferring between relational domains falls under heterogeneous transfer learning, as the feature spaces differ. In relational domains, the feature space is defined by the domain vocabulary. By vocabulary, we mean the set of predicates and term types used to describe instances in the domain.

Transfer Strategies. One of the key questions in designing a transfer learning algorithm is "*how to transfer?*". In what follows, we present different transfer approaches to address this question.

- **Feature-based transfer learning.** Involves projecting the source and target data into a shared feature space to reuse the labeled data from the source domain. The key challenge lies in learning a new feature space of high quality. This category is recommended when there is little feature overlap between the source and target domains.

- **Model-based transfer learning.** Reuse a pre-trained model learned from the source domain. Typically, the target model is initialized with the pre-trained model's parameters and is then fine-tuned using a small set of labeled data from the target domain.

- **Relation-based transfer learning.** It is a more recent approach to transfer learning. It assumes that certain relationships between objects are common to both domains. Therefore, it transfers rules learned for these relationships in the source to the target domain.

- **Instance-based transfer learning.** Assumes that reusing labeled instances from the source domain can effectively support learning in the target domain. To reuse source instances successfully, these methods usually rely on an instance reweighting or resampling strategy. This category is recommended when the source and target feature spaces significantly overlap.

These categories for transfer strategies are not mutually exclusive. For example, our work is built mainly on the instance-based category. However, it also fits into the relation-based category, as our key motivation for reusing instances from the source domain to the target domain is the expectation that both of them share similarities in how objects are related.

Negative Transfer Regardless of the transfer strategy, all transfer learning methods should aim to avoid transferring inappropriate or irrelevant knowledge from the source domain to the target domain. Failing to do so can result in models that perform worse than those learned from scratch, a phenomenon known as *negative transfer* [46]. Negative transfer is closely related to two key challenges in designing a transfer learning algorithm: *when to transfer* and *what to transfer*.

The first challenge concerns whether the transfer will benefit the learning process. This is relevant when choosing the source domains from which knowledge will be transferred. If the source domain is not closely related to the target domain, the transfer may lead to negative transfer. Thus, one mechanism to mitigate negative transfer is selecting an appropriate source domain [40, 47]. The second challenge focuses on identifying which parts of the source domain knowledge can effectively support learning in the target domain. For example, in instance-based transfer learning, this could involve determining which instances are relevant to transfer [8, 34].

Utility-driven approach to instance selection. The method proposed in this work is based on instance-based transfer learning [8, 9, 34]. The goal is to reuse instances from a source domain to improve data quantity and quality before training a model for the task of interest in the target domain. Instead of just concatenating the datasets, we use the concept of utilities to identify relevant instances. Specifically, the instance selection problem is framed as a fair resource allocation problem, where the model's predictive capability is allocated to instances based on their relevance to learning. We hypothesize that this approach can help mitigate the negative transfer effect and improve the performance of the resulting models.

3.3 Relational Regression Trees

Motivated by the success of decision trees in the propositional domain, BLOCKEEL e DE RAEDT [29] introduced their relational extension, First-Order Logical Decision Trees (FOLDTs), and adapted the C4.5 algorithm [48] to induct FOLDTs. RRTs are the regression counterpart of FOLDTs and are considered an ILP model. In the

literature, this model is commonly referred to as TILDE, which denotes both the learning model and the system that implements learning and inference for FOLDTs.

Structure of a FOLDT. Figure 3.1 illustrates a FOLDT to predict the relation *sister_of/2* from the domain described in Table 3.1. Unlike traditional decision trees, a FOLDT is necessarily a binary tree in which branches indicate the success (left branch) or failure (right branch) of tests at internal nodes. The internal nodes, represented by ellipses, contain conjunctions of logical literals rather than attributes or attribute-value pairs. Additionally, these nodes may share logical variables, but a variable introduced in an internal node must not appear on its right branch. This is because the right subtree of a node is only relevant when the test performed on that node fails. Since each variable introduced in a node is existentially quantified, it follows that there is no such value for the recently introduced variable that makes the example succeed in the test, and it should not be referred to again in the subtree.

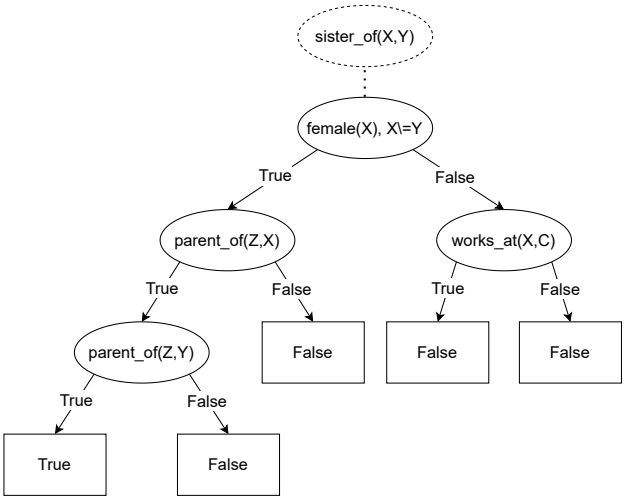


Figure 3.1: A suboptimal first-order logical decision tree encoding the relation *sister_of/2* of Table 3.1.

Performing inference in FOLDT. Inferences with a FOLDT are performed by traversing the tree from the root node to a leaf, using the facts describing an example and the tests at internal nodes. Due to the dependencies among variables in different internal nodes, a test at a specific node is based on a query that combines the conjunctions of literals from higher-level nodes, where the test succeeds for the example, along with the conjunction of literals from the current node. Depending on the result of the test, the example is sorted to either the left (success) or the right (failure) subtree, continuing this process until a leaf is reached. Leaves, represented by rectangles, provide the predictions, which may be either a regression value, as in

RRTs, or a class label, as in TILDE for classification. Figure 3.1 is an example of a TILDE for classification tasks.

Learning a FOLDT. A FOLDT is learned in a recursive process that starts with an empty tree and progressively finds the best split for each internal node based on the training examples reaching that node and a predefined split criterion, such as entropy or weighted variance. The examples used to build the tree consist of logical facts describing instances, similar to those in Table 3.1. The tree stops growing when examples reaching a node are insufficient to be split into subtrees or when it meets a stopping condition, such as the maximum tree depth.

The search space in relational domains is very large. To reduce this space and make learning feasible, the set of candidate conjunctions of literals to be evaluated as tests for splitting a node is determined by a refinement operator. This operator generates a set of specializations of the query associated with the node where the split is performed by adding new literals according to user specifications, similar to the PROGOL system [28]. These specifications, consisting of mode declarations, guide the search by indicating which literals can be added as refinements of a query, which variables can appear in those literals, how variable types should be bound, and how many times the same predicate can be used in the query.

An example of mode declarations is provided in the last column of Table 3.1. The literals listed there are those allowed to be added to a query by the refinement operator when generating candidate conjunctions of literals to form the new internal nodes. Furthermore, each term is associated with a type and a symbol "+", "−", or "#", indicating the type of the term in the new literal and whether it should be an input variable, output variable, or constant, respectively.

FOLDTs are deterministic. Due to their deterministic nature, we do not use RRT directly in this work. Instead, we rely on RDN-Boost, an approach to learning RDNs that employs RRTs to capture complex and uncertain dependencies among logical atoms. The details of this approach are discussed in the following section.

3.4 Boosted Relational Dependency Networks

In this section, we introduce RDNs followed by a boosting algorithm to learn those networks.

Relational Dependency Networks

Relational Dependency Networks (RDN) [31] extends *Dependency Networks (DNs)* [49] to the relational domain. Instead of using the original formulation based

on relational databases, this work adopts the logical perspective of RDNs as introduced in [33]. Based on that, an RDN is defined as an approximate model in which the joint probability distribution of a set of logical atoms is modeled as the product of their conditional probability distributions (CPDs), which can be learned independently. Thanks to this, RDNs present features that make them more interesting compared to other related models, including interpretability and efficient learning of the model's parameters and structure.

As in DNs, RDNs are formalized as a directed graph $G(V, E)$ whose nodes $v_\ell \in V$ represent atoms π_ℓ containing logical variables (e.g., *works_at(Person, Company)*) that can be instantiated to generate *ground atoms* (e.g., *works_at(ron, google)*), and edge $e_{m\ell} \in E$ encodes the dependency between atoms π_m and π_ℓ. We denote by $\mathbf{MB}(\pi_\ell)$ the Markov blanket of π_ℓ, i.e., the set of atoms that make π_ℓ conditionally independent to other atoms, $P(\pi_\ell \mid \mathbf{\Pi} \setminus \{\pi_\ell\}) = P(\pi_\ell \mid \mathbf{MB}(\pi_\ell))$, where $\mathbf{\Pi}$ denotes the set of all atoms. Note that an edge $e_{m\ell} \in E$ implies that $\pi_m \in \mathbf{MB}(\pi_\ell)$. Each node is associated with a CPD of the form $P(\pi_\ell \mid \mathbf{MB}(\pi_\ell))$ and captures the probability of an atom π_ℓ given the set of all atoms on which it depends ($\mathbf{MB}(\pi_\ell)$).

Learning an RDN is the same as learning the CPDs. Although learning CPDs was initially addressed with relational probability trees [31], the approach based on functional gradient boosting using RRTs introduced in [33] proved to be more efficient. For this reason, we dedicate a more detailed description of this learning method, also known as Boosted Relational Dependency Networks (RDN-Boost).

Functional Gradient Boosting

Let $D_\pi = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a relational dataset relative to the target predicate π. Each instance $i \in D_\pi$ represents a negative or a positive sample of a predicate π and is characterized by $\mathbf{x}_i$, the set of facts that influence it (which may also influence other instances), and by $y_i \in \{0, 1\}$, the label associated with it. Recall from the previous section that $\mathbf{MB}(\pi)$ represents the set of all atoms that π directly depends on. Thus, $\mathbf{x}_i$ represents the observed values of the atoms influencing π (i.e., of $\mathbf{MB}(\pi)$) and y_i represents the observed value of the target predicate π. As pointed in the previous section, observed values of atoms are also referred to as *ground atoms*.

We define the model inspired by the framework of logistic regression, for binary classification. Let Y_i be a random variable, produced by the model, representing whether the target predicate is predicted to be positive for the i-th instance. The model estimates the probability of the target predicate π being positive (i.e., $Y_i = 1$) based on the facts $\mathbf{x}_i$, applying a sigmoid function to a learnable function ψ.

The learnable function is given by $\psi : \mathbf{X} \to \mathbb{R}$, where $\mathbf{X}$ is the space of all

possible groundings of predicates in $\mathbf{MB}(\pi)$. For each instance i, we have

$$P(Y_i = 1 \mid \mathbf{x}_i; \psi) = \frac{1}{1 + e^{-\psi(\mathbf{x}_i)}}. \tag{3.1}$$

Moreover, letting

$$\phi(\mathbf{x}_i) = P(Y_i = 1 \mid \mathbf{x}_i; \psi) \tag{3.2}$$

we can express the likelihood of observing y_i given the facts $\mathbf{x}_i$ as:

$$P(y_i \mid \mathbf{x}_i; \psi) = \phi^{y_i}(\mathbf{x}_i)(1 - \phi(\mathbf{x}_i))^{1-y_i} = \begin{cases} \phi(\mathbf{x}_i) & \text{if } y_i = 1 \\ 1 - \phi(\mathbf{x}_i) & \text{if } y_i = 0. \end{cases} \tag{3.3}$$

Learning the desired function ψ is achieved by optimizing an objective function, which normally is the conditional log-likelihood (CLL) of the dataset D_π, given by

$$\mathcal{L}(D_\pi; \psi) = \ln \prod_{i=1}^{N} P(y_i \mid \mathbf{x}_i; \psi) = \sum_{i=1}^{N} y_i \ln \phi(\mathbf{x}_i) + (1 - y_i) \ln(1 - \phi(\mathbf{x}_i)). \tag{3.4}$$

Under the objective of maximizing the above function, we will argue that its partial derivatives relative to improvements on ψ is $y_i - \phi(\mathbf{x}_i)$.

An iterative two-stage process to determine ψ. Next, we aim to determine ψ, implicitly defined by Eq. (3.1). Unlike the case of logistic regression, ψ is a nonparametric function. Following the functional gradient boosting method, we approximate ψ by the sum of K regression functions, also known as regressors, for some $K > 0$. The regressors are computed iteratively and incrementally, and for $k \in \{1, \ldots, K\}$ the k-th iteration produces both the increment Δ_k and the regressor ψ_k, as in

$$\psi_k = \psi_0 + \Delta_1 + \cdots + \Delta_k, \tag{3.5}$$

where $\psi_k : \mathbf{X} \to \mathbb{R}$.

Stage 1. Under the objective of maximizing Eq. (3.4), the partial derivative of the objective function with respect to $\psi_{k-1}(\mathbf{x}_i)$, at the i-th instance, is given by

$$\delta_{k,i} = \frac{\partial \mathcal{L}(\{(\mathbf{x}_i, y_i)\}; \psi_{k-1})}{\partial \psi_{k-1}(\mathbf{x}_i)} \tag{3.6}$$

$$= \frac{\partial (y_i \ln \phi_{k-1}(\mathbf{x}_i) + (1 - y_i) \ln(1 - \phi_{k-1}(\mathbf{x}_i)))}{\partial \psi_{k-1}(\mathbf{x}_i)} = y_i - \phi_{k-1}(\mathbf{x}_i), \tag{3.7}$$

where

$$\phi_{k-1}(\mathbf{x}_i) = P(Y_i = 1 \mid \mathbf{x}_i; \psi_{k-1}) = \frac{1}{1 + e^{-\psi_{k-1}(\mathbf{x}_i)}}. \tag{3.8}$$

Equation (3.7) corresponds to the difference between the actual label for each data point and the model prediction. Such difference is the error, used to update the model in the next iteration, allowing the model to gradually become more accurate.

At each iteration k, the above parameters are computed using instance-based computations, capturing the specific relationships between features and the target variable for every data point. Once these parameters are determined at the instance level, as indicated above, they are generalized using RRTs [29, 33], an extension of regression trees to the relational domain, as detailed in the sequel.

Stage 2. Let D_δ be the collection of available samples produced at stage 1, $D_\delta = \{((\mathbf{x}_i, y_i), \delta_{k,i})\}_{i=1}^N$, where $\delta_{k,i}$ is given by Eq. (3.7). We denote by $\hat{h}_k$ the regressor to the samples in D_δ. Typically, $\hat{h}_k$ consists of a RRT and is obtained by minimizing the squared error between its estimates and $\delta_{k,i}$,

$$\hat{h}_k = \text{argmin}_h \sum_{i=1}^{N} \left(h(\mathbf{x}_i, y_i) - \delta_{k,i}\right)^2 .$$

Then,

$$\psi_k = \psi_0 + \eta_1 \hat{h}_1 + \cdots + \eta_k \hat{h}_k, \tag{3.9}$$

where η_k is the learning rate at the k-th step, and $\Delta_k = \eta_k \hat{h}_k$ (see Eq. (3.5)). In the realm of boosted models, the RRT $\hat{h}_k$ is also referred to as a *weak learner*, that is fitted to the error at the k-th iteration. The model is built iteratively, where each weak learner focuses on the errors made by the previous one.

In summary, Eq. (3.7) computes the error at each step, while Eq. (3.9) uses this error information to iteratively update and improve the model. The model becomes more accurate over time by learning from its mistakes at each step, improving its performance gradually by focusing more on difficult cases.

After the iterative process is concluded at step K, we let ψ in Eq. (3.1) be approximated by ψ_K. This iterative process ensures that the model, at every step, balances detailed instance-level accuracy (stage 1) with the ability to generalize across diverse data points (stage 2), as further detailed in [50–55]. Stages 1 and 2 are summarized in Algorithm 1.

Accounting for utilities and multiple domains. In the remainder of this work, we will indicate how to adapt the above framework to account for utilities and multiple domains, such as a source and a target domain in the realm of transfer learning. In particular, we will generalize Eq. (3.4) and then show how to adapt Eq. (3.7) accordingly.

Algorithm 1 Learn $P(\pi \mid \mathbf{MB}(\pi))$ using RDN-Boost

Require: Initial guess ψ_0, dataset D_π, number of iterations K, and learning rates η_k

Ensure: The CPD $P(\pi \mid \mathbf{MB}(\pi))$ for the target predicate π

```
for k = 1 to K do
    D_δ ← ∅                                          ▷ Stage 1 begins here
    for i ∈ D_π do
        δ_{k,i} ← y_i − φ_{k−1}(x_i)                 ▷ See Eq. (3.7)
        D_δ ← D_δ ∪ {((x_i, y_i), δ_{k,i})}
    end for
    ĥ_k ← Fit a RRT from D_δ                         ▷ Stage 2 begins here
    ψ_k ← ψ_{k−1} + η_k ĥ_k
end for
return φ_K                                           ▷ φ_K is a function of ψ_K, see Eq. (3.8)
```

3.5 Utility Maximization

Utility maximization is a mathematical tool widely used to address the problem of *fair resource allocation* in the realm of Economics, Computer Networks [10, 56] and Federated Machine Learning [12]. In that task, R units of a given *resource* (e.g., link bandwidth) should be allocated to N *agents* (e.g., users). The total of resources allocated to the i-th agent is denoted by $r_i \geq 0$ and the user's level of satisfaction with this allocation is measured by $u(r_i)$, where $u : \mathbb{R} \to \mathbb{R}$ is called a *utility function*. A utility function is required to be a continuously differentiable, increasing and concave function. Given the allocation $\mathbf{r} = (r_1, \ldots, r_N) \in \mathbb{R}^N$, the *total satisfaction* is defined as

$$U(\mathbf{r}) = \sum_{i=1}^{N} w_i u(r_i), \tag{3.10}$$

where w_i is the weight associated with the i-th agent. The goal of utility maximization is to solve the following optimization problem:

$$\max_{\mathbf{r}} \quad U(\mathbf{r}) \quad \text{s.t.} \quad \sum_{i=1}^{N} r_i \leq R. \tag{3.11}$$

We provide further details on how we use the utility maximization framework in Chapter 4. Broadly speaking, in our instance-based approach to transfer learning, each agent represents a data point (instance) from a domain of interest. The learner should learn how to allocate its predictive capacity to instances. In particular, we consider r_i as the predicted probability mass associated with the actual class of the instance. Additionally, we tackle the optimization problem without explicit constraints, given our choice of model design. Finally, we rely on α-fairness [15, 16].

Utility functions usually allow for the implementation of different notions of *fairness*. In particular, often three definitions of fairness stand out, namely: *max-throughput fairness* (which in [12, 57] is referred to as *utilitarianism*), *proportional fairness* [10], and *max-min fairness* (referred to as *egalitarianism* in [12, 57]). All these variations are encompassed by α-*fairness*, defined as

$$u(x) = \begin{cases} \frac{x^{1-\alpha}}{1-\alpha} & \text{if } \alpha \neq 1 \\ \ln x & \text{if } \alpha = 1. \end{cases} \tag{3.12}$$

Chapter 4

Methodology

This chapter outlines our methodology. We begin by formalizing the problem in Section 4.1. Section 4.2 introduces our general framework for instance-based transfer learning in statistical relational learning, applying it to RDN-Boost. We discuss source and target datasets preprocessing in Section 4.3. Section 4.4 introduces the use of utilities in the learning process of RDN-Boost and extends it to account for multiple domains. Next, we propose an instance weighting strategy to mitigate strong imbalance among groups of instances in Section 4.5.

4.1 Problem Statement

In this work, we leverage instance-based transfer learning to enhance the performance of a predictive model in a target relational domain where training data are limited. This strategy involves reusing instances from a source dataset sampled from another relational domain to build a target model. This requires instances from different domains to be expressed in a common vocabulary. Next, we can employ our augmented version of RDN-Boost to learn from multiple domains and identify which instances benefit or negatively impact the model.

4.2 UTIL-BRDN

We propose UTIL-BRDN, a utility-driven approach to transfer instances for learning RDN-Boost. It comprises two stages, as illustrated in Figure 4.1. The first stage combines instances from two distinct relational domains into a single dataset, all mapped to the vocabulary of the target domain. Then, a predictive model is fitted from the combined dataset instances. Using source and target data to train a model, we expect to improve the performance of the target task compared to learning from scratch. To achieve this, we must prevent negative transfer. This is tackled in the

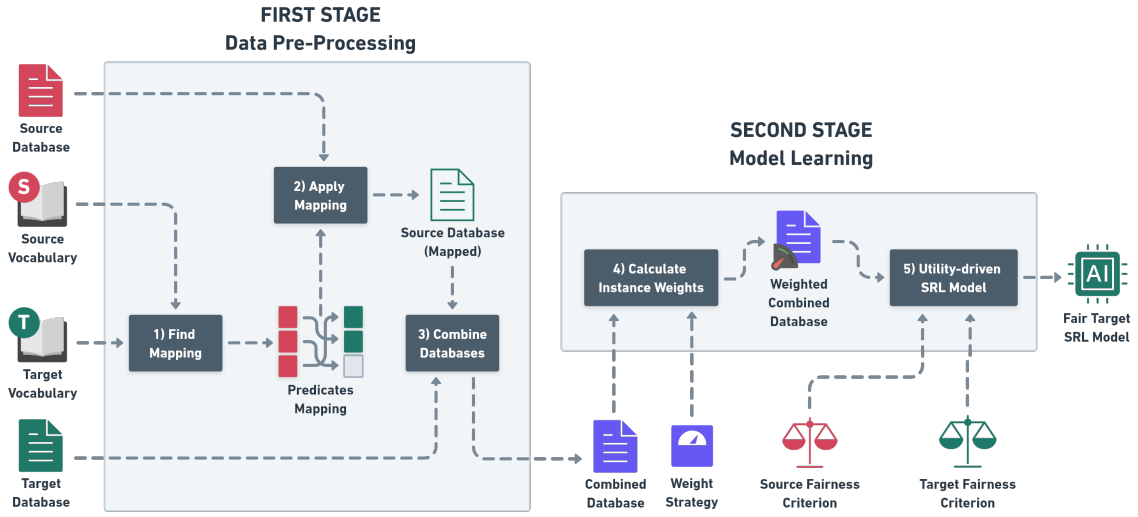


Figure 4.1: Utility-driven approach to Transfer Instances for Learning Boosted Relational Dependency Networks (UTIL-BRDN).

second stage, where we use an instance selection mechanism to prevent transferring poor instances. This mechanism is inspired by the problem of fair resource allocation. UTIL-BRDN [1] is built on the implementations of BoostSRL[2] and on srlearn[3]. In the following sections, we discuss each stage in more detail.

4.3 Combining Instances from Source and Target Domains

This section describes steps 1-3 of Figure 4.1. We use the term vocabulary to refer to the high-level data structure describing which entities, relationships, and attributes constitute the domain of interest. From the perspective of First-Order Logic, a vocabulary consists of a set of *template atoms*, each of which corresponds to a predicate from the domain applied to its term types instead of objects. The difference between a regular atom and a template atom relies on terms, which in regular atoms represent objects in the domain and in template atoms correspond to object types (entities). Therefore, one needs to map the vocabularies appropriately before integrating the datasets. Compared with the propositional case, mapping vocabularies is equivalent to reducing instances from different domains to the same feature space.

To establish a mapping from source instances to the target domain, the first

[1] UTIL-BRDN is available on GitHub: `https://github.com/cainafigueiredo/util-brdn`.

[2] BoostSRL is a Java-based system specialized in SRL models powered by gradient-based boosting, including RDN-Boost.

[3] srlearn is a Python package which provides an interface based on scikit-learn for BoostSRL models.

step involves mapping their respective vocabularies, i.e., finding a correspondence between the predicates and the term types from the source and target domains. A mapping from a source template atom $p(s_1, \ldots, s_n)$ to a target template atom $q(t_1, \ldots, t_n)$ is denoted as

$$M_{p \mapsto q} = (p/n \mapsto q/n, \{s_1 \mapsto t_1, \ldots, s_n \mapsto t_n\}) \,. \tag{4.1}$$

In this expression, the first element of the tuple specifies the mapping between two predicates of the same arity, while the second one comprises the mapping of their respective term types. A source atom can also be mapped to an "empty" atom, e.g., when it lacks a corresponding target atom of the same arity.

As source atoms are mapped, it is imperative to maintain consistency across all term mappings. For instance, if $s_1 \mapsto t_1$ after performing $M_{p \mapsto q}$, then subsequent mappings such as $M_{p' \mapsto q'} = (p'/1 \mapsto q'/1, \{s_1 \mapsto t_2\})$, with $t_2 \neq t_1$, are prohibited, as $s_1 \mapsto t_2$ contradicts the prior assignment. A mapping is then said to be *legal* if all source atoms are mapped either to a valid target atom or an empty atom, without violating global term mappings.

Moreover, except for mappings to the empty atom, we enforce one-to-one predicate mappings, similarly to what is done in [6, 7]. This means that each source predicate is mapped to a single target predicate (or to the empty predicate), and vice-versa. Unlike previous work, which typically requires a match between the target predicates from the source and target domains to transfer the structure of a pre-trained model, our approach does not force such a mapping. Instead, any predicate from the source domain can be mapped to the target predicate in the target domain, provided they are compatible with each other, which provides greater flexibility in finding higher-quality mappings. However, this flexibility also introduces increased complexity due to the expanded search space.

Algorithm 2 outlines the process of generating all valid mappings under these assumptions. The procedure *FindAllValidMappings* serves as an initialization step to ensure that the mappings always map a predicate from the source vocabulary to the target predicate in the target vocabulary. This ensures that new positive and negative instances for the target predicate are added to the dataset, allowing the UTIL-BRDN objective function to account for these new instances in the learning process. Then, *ExpandMappings* is called recursively until all predicates in the source vocabulary are mapped to predicates in the target vocabulary.

After finding a legal mapping between the vocabularies, all source instances are mapped to the target domain and concatenated to the target dataset. To prevent collisions between constants from the mapped and target instances, we prefix all constants from the mapped instances with the word *source*. In addition, we track

the origin of each example within the combined dataset to effectively control its impact on the final model.

Mapping Selection Strategy. Multiple legal mappings may exist between a pair of vocabularies. The challenge of identifying the best mapping has been addressed in previous work (see Section 2.3). However, most of these approaches consist of greedy methods relying on pre-trained models to build the mappings, making them unsuitable for our instance-based transfer learning method, which leverages source data instead of a pre-trained model.

This work does not aim to adapt existing mapping search approaches or to develop new methods. Instead, we introduce an instance-based transfer learning method within the context of statistical relational learning and empirically analyze its properties. Future research could extend our approach to propose new algorithms that enhance the mapping process.

In our experiments, we employ a simple mapping selection method based on random sampling, where a mapping is uniformly selected from the set of all legal mappings generated by Algorithm 2. While random search can be inefficient and may struggle to find optimal solutions, it is often used as a baseline in search problems, providing a valuable comparison point for future strategies associated with UTIL-BRDN to enhance mapping selection. Additionally, this straightforward approach allows us to gain meaningful insights into transfer learning using UTIL-BRDN.

4.4 Augmenting RDN-Boost to Account for Multiple Domains

RDN-Boost was originally designed to learn a RDN from a single domain. To enhance its capability to utilize data from another domain when learning a target task, one could preprocess the data using our instance combination strategy introduced in the previous section. Then, the combined data can be fed into the original RDN-Boost algorithm described in Section 3.4. However, this straightforward approach does not assess the relevance of instances from different domains, which can lead to irrelevant or low-quality instances negatively impacting the model's performance.

To address this issue, we introduce steps 4 and 5 in Figure 4.1. More details about step 4, which involves instance weighting, are provided in Section 4.5. In what follows, we focus on step 5, where we introduce a modified version of RDN-Boost. In this version, the CLL maximization used to learn the RDN in the original RDN-Boost is replaced by an alternative optimization problem described in Section 4.4.2. We also illustrate how our approach generalizes the original RDN-Boost. Specifically,

our novel objective function integrates instances from both the source and target domains and is built on the utility maximization formalism. Rather than introducing our final proposal immediately, we first demonstrate how to incorporate utilities into a single-domain scenario in Section 4.4.1. Then, in Section 4.4.2, we explicitly extend our approach to simultaneously account for both the source and target domains.

4.4.1 Incorporating Utilities into RDN-Boost

Let

$$z_i = P(y_i \mid \mathbf{x}_i; \psi) = \begin{cases} \phi(\mathbf{x}_i) & \text{if } y_i = 1 \\ 1 - \phi(\mathbf{x}_i) & \text{if } y_i = 0 \end{cases} \tag{4.2}$$

be the probability mass function evaluated at instance $i = 1, \ldots, N$. In the same spirit that led from the definition in Eq. (3.3) to the objective function in Eq. (3.4), we use the definition of z_i in Eq. (4.2) to define our new objective function, which is an adaptation of Eq. (3.10) to our approach to instance-based transfer learning. This adaptation consists in substituting $\mathbf{z} = (z_1, \ldots, z_N)$ for $\mathbf{r}$ and results in

$$\mathcal{U}(D_\pi; \psi) = U(\mathbf{z}) = \sum_{i=1}^{N} w_i u(z_i). \tag{4.3}$$

Consequently, the partial derivative in Eq. (3.7) becomes

$$\delta_{k,i} = \frac{\partial \mathcal{U}(\{(\mathbf{x}_i, y_i)\}; \psi_{k-1})}{\partial \psi_{k-1}(\mathbf{x}_i)} \tag{4.4}$$

$$= w_i \frac{\partial u(z_i)}{\partial \psi_{k-1}(\mathbf{x}_i)} \tag{4.5}$$

$$= w_i \frac{\partial u(z_i)}{\partial z_i} \frac{\partial z_i}{\partial \phi_{k-1}(\mathbf{x}_i)} \frac{\partial \phi_{k-1}(\mathbf{x}_i)}{\partial \psi_{k-1}(\mathbf{x}_i)} \tag{4.6}$$

$$= w_i z_i^{1-\alpha} (y_i - \phi_{k-1}(\mathbf{x}_i)). \tag{4.7}$$

The derivatives appearing in Eq. (4.7) are as follows:

$$\frac{\partial u(z_i)}{\partial z_i} = \begin{cases} \frac{\partial}{\partial z_i}\left(\frac{z_i^{1-\alpha}}{1-\alpha}\right) & \text{if } \alpha \neq 1 \\ \frac{\partial}{\partial z_i}\ln(z_i) & \text{if } \alpha = 1 \end{cases} \tag{4.8}$$

$$= \begin{cases} z_i^{-\alpha} & \text{if } \alpha \neq 1 \\ z_i^{-1} & \text{if } \alpha = 1 \end{cases} \tag{4.9}$$

$$= z_i^{-\alpha}, \tag{4.10}$$

$$\frac{\partial z_i}{\partial \phi_{k-1}(\mathbf{x}_i)} = \begin{cases} \frac{\partial \phi_{k-1}(\mathbf{x}_i)}{\partial \phi_{k-1}(\mathbf{x}_i)} & \text{if } y_i = 1 \\ \frac{\partial (1-\phi_{k-1}(\mathbf{x}_i))}{\partial \phi_{k-1}(\mathbf{x}_i)} & \text{if } y_i = 0 \end{cases} \tag{4.11}$$

$$= (-1)^{1-y_i}, \tag{4.12}$$

$$\frac{\partial \phi_{k-1}(\mathbf{x}_i)}{\partial \psi_{k-1}(\mathbf{x}_i)} = \phi_{k-1}(\mathbf{x}_i)(1-\phi_{k-1}(\mathbf{x}_i)). \tag{4.13}$$

Putting them together, we can proceed with the calculation of $\delta_{k,i}$:

$$\delta_{k,i} = w_i z_i^{-\alpha}(-1)^{1-y_i}\phi_{k-1}(\mathbf{x}_i)(1-\phi_{k-1}(\mathbf{x}_i)) \tag{4.14}$$

$$= w_i(y_i - \phi_{k-1}(\mathbf{x}_i))(\phi_{k-1}^{1-\alpha}(\mathbf{x}_i))^{y_i}((1-\phi_{k-1}(\mathbf{x}_i))^{1-\alpha})^{1-y_i} \tag{4.15}$$

$$= w_i(y_i - \phi_{k-1}(\mathbf{x}_i))(\phi_{k-1}^{y_i}(\mathbf{x}_i)(1-\phi_{k-1}(\mathbf{x}_i))^{1-y_i})^{1-\alpha} \tag{4.16}$$

$$= w_i z_i^{1-\alpha}(y_i - \phi_{k-1}(\mathbf{x}_i)). \tag{4.17}$$

This expression for $\delta_{k,i}$ allows us to use RDN-Boost to learn a model under the alternative objective function given in Eq. (4.3). In Section 4.4.2, we augment the model to simultaneously accommodate instances from both a source and a target domain.

Derivatives, fairness, and instance selection. In Eq. (4.17), the derivative $\delta_{k,i}$ for iteration k and instance i corresponds to the prediction error $y_i - \phi_{k-1}(\mathbf{x}_i)$ modulated by the factor $w_i z_i^{1-\alpha}$. This factor is influenced by the instance weight w_i, the prediction $\phi_{k-1}(\mathbf{x}_i)$, the instance class y_i, and the modulation parameter α. Figure 4.2 illustrates how different values of α implicitly affect the derivatives and, consequently, the relevance of each instance for learning. Solid lines represent positive instances ($y_i = 1$), dashed lines represent negative instances ($y_i = 0$), and we consider $w_i = 1$. The derivative captures different notions of fairness depending

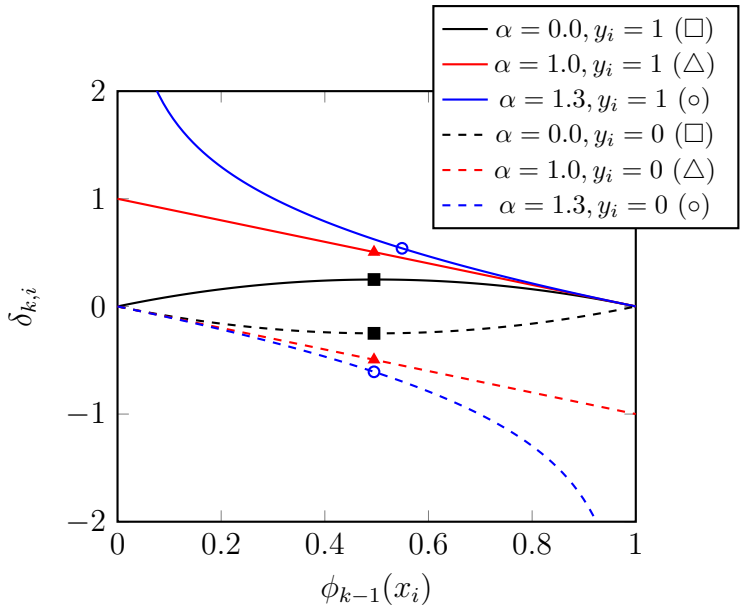


Figure 4.2: Derivative associated with the instance i at iteration k for our utility-driven version of RDN-Boost, letting $w_i = 1$.

on the value of α:

- **Throughput-Maximization Fairness**: $\alpha = 0$, and more generally any $\alpha \in [0, 1)$, reduces the influence of instances with large prediction errors by decreasing the absolute value of their derivatives compared to the original RDN-Boost. The smaller the value of α, the less significant these instances become. In the context of transfer learning, which will be discussed in the next section, this approach is justified by the fact that source domain instances that are difficult to predict in the target domain may not actually be relevant for the latter [9].

- **Max-Min Fairness**: $\alpha = 1.3$, and more generally any $\alpha \in (1, \infty)$, amplifies the influence of instances with large prediction errors by increasing the absolute value of their derivatives compared to RDN-Boost. This causes the model to focus more on these instances in the subsequent learning iteration.

- **Proportional Fairness**: for $\alpha = 1$, the original RDN-Boost becomes a special case of our method, as by Eq. (3.7), the derivatives of RDN-Boost and the proposed approach become equivalent when setting $\alpha = 1$ and $w_i = 1$ for every instance i.

4.4.2 Integrating Utilities From Source and Target Domains

By incorporating utilities into the optimization objective of RDN-Boost, we obtained a mechanism to determine the relevance of each instance during the learning process.

In particular, the weight w_i and the term $z_i^{1-\alpha}$ in Eq. (4.17) provide a handle on instance selection relative to i (an explicit one in the former case, an implicit one in the latter), in the form of instance reweighting. This is of particular interest to instance-based transfer learning, but it cannot distinguish between instances from different domains.

There are several reasons to change model behavior based on the instance domain. For example, one could expect to perform better in the target task by tuning the model to improve its predictions for misclassified instances in the target domain while ignoring instances with poor predictions in the source domain. This would be based on assuming that the source domain has several noisy instances, and, consequently, on trying to filter them out based on how hard it is to predict their classes using the partially fitted model. Likewise, it would attempt to reduce the prediction error for instances in the target domain, assuming that such instances are important and that the model needs to learn novel patterns to accommodate them. This sort of approach is followed by TrAdaBoost [9], which does transfer learning on AdaBoost.

Let $D_\pi = D_{\pi,\mathrm{S}} \cup D_{\pi,\mathrm{T}}$ be a dataset for target predicate π combining source instances $D_{\pi,\mathrm{S}}$ mapped to the target vocabulary and target instances $D_{\pi,\mathrm{T}}$. Then, $\tau(i) \in \{\mathrm{S}, \mathrm{T}\}$ denotes the domain (source or target, respectively) of the i-th instance,

$$\tau(i) = \begin{cases} \mathrm{S} & \text{if } i \in D_{\pi,\mathrm{S}} \\ \mathrm{T} & \text{if } i \in D_{\pi,\mathrm{T}}. \end{cases} \tag{4.18}$$

We extend the objective function in Eq. (4.3) to explicitly account for both domains:

$$\mathcal{U}(D_\pi; \psi) = \sum_{i \in D_{\pi,\mathrm{S}}} w_i u_{\mathrm{S}}(z_i) + \sum_{i \in D_{\pi,\mathrm{T}}} w_i u_{\mathrm{T}}(z_i) = \sum_{i \in D_\pi} w_i u_{\tau(i)}(z_i), \tag{4.19}$$

where $u_{\tau(i)}(\cdot)$ corresponds to the α-fairness utility function with $\alpha = \alpha_{\tau(i)}$.

The partial derivative associated with this new objective function is then calculated as follows:

$$\delta_{k,i} = \frac{\partial \mathcal{U}(\{(\mathbf{x}_i, y_i)\}; \psi_{k-1})}{\partial \psi_{k-1}(\mathbf{x}_i)} = w_i \frac{\partial u_{\tau(i)}(z_i)}{\partial \psi_{k-1}(\mathbf{x}_i)} = w_i z_i^{1-\alpha_{\tau(i)}} (y_i - \phi_{k-1}(\mathbf{x}_i)). \tag{4.20}$$

Note that the above partial derivative can be derived from Eq. (4.7), replacing α by its domain-specific counterpart, $\alpha_{\tau(i)}$.

This new expression for $\delta_{k,i}$ enables us to use RDN-Boost to learn a model under the multi-domain, utility-driven objective function in Eq. (4.19). Unlike CLL, our method to learn RDNs extends RDN-Boost to multiple domains and provides a flexible mechanism to determine the relevance of each instance, accounting explicitly for its domain. In particular, α_{S} and α_{T} indicate whether the model should

improve its performance on misclassified instances or just filter them out by decreasing their derivatives according to their domain. For example, a heuristic similar to TrAdaBoost [9] can be achieved by selecting $\alpha_{\mathrm{S}} \in [0, 1)$ and $\alpha_{\mathrm{T}} \in (1, \infty)$.

Warm-up iterations. We introduce an additional hyperparameter $K_{\mathrm{w}} \in \{1, \ldots, K\}$ to indicate the number of so-called *warm-up iterations*. To motivate this, consider the term $z_i^{1-\alpha_{\tau(i)}}$ in Eq. (4.20). This term modulates the original partial derivative from RDN-Boost, given by $y_i - \phi_{k-1}(\mathbf{x}_i)$. Its value depends on the prediction corresponding to instance i, given by $\phi_{k-1}(\mathbf{x}_i)$, noting that z_i is defined by Eq. (4.2). During the earliest learning iterations, z_i is likely uninformative and, as such, may not warrant being used for modulation purposes. To mitigate this issue, we set $\alpha_{\mathrm{S}} = 1$ and $\alpha_{\mathrm{T}} = 1$ during iterations $k = 1, \ldots, K_{\mathrm{w}}$. The remaining iterations employ the user-defined α_{S} and α_{T} values.

Learning algorithm. Algorithm 3 extends Algorithm 1 to accommodate our proposed multi-domain, utility-driven version of RDN-Boost. Whereas Algorithm 1 can only learn from single-domain datasets, the dataset D_π in Algorithm 3 comprises instances from both the source and target domains, allowing it to learn from both. While doing so, the computation of $\delta_{k,i}$ in Algorithm 3 accounts for utilities, which modulate the impact of each instance on the final model.

Approaches to set α-values. Our multi-domain, utility-driven version of RDN-Boost is inspired by the literature on utility maximization and fair resource allocation. Consequently, setting α_{S} and α_{T} can be interpreted as choosing a fairness criterion for each of the domains. This provides the user with a tool that can be leveraged for two purposes: to learn a fair model or to optimize the model's predictive performance.

In the first scenario, the goal is to learn a model that meets a specific fairness criterion. The model designer sets the values of α_{S} and α_{T} to achieve this criterion, prioritizing fairness over maximal accuracy. Thus, the chosen model may not be the most accurate overall, but it aligns with the desired fairness criterion, which is enough. In the second scenario, the designer tunes the hyperparameters to achieve the most accurate model possible, for example through grid search. By evaluating different values of α_{S} and α_{T}, the designer identifies a fairness regime where the model achieves the highest accuracy within the explored search space. In other words, the utility function used as the optimization objective to obtain the most accurate model is selected after considering a range of fairness criteria. In this work, we explore the latter strategy to set the α-values.

Table 4.1: Use cases for UTIL-BRDN

Source Data	**RDN-Boost Version**	**Transfer**	**Instance Selection**
no	original ($\alpha_S = 1, \alpha_T = 1, w_i = 1$)	no	no
no	multi-domain, utility-driven	no	yes
yes	original ($\alpha_S = 1, \alpha_T = 1, w_i = 1$)	yes	no
yes	multi-domain, utility-driven	yes	yes

4.4.3 Use Cases for UTIL-BRDN

Table 4.1 summarizes the different learning scenarios that can be addressed by UTIL-BRDN. When no source data is provided, UTIL-BRDN performs learning from scratch. In contrast, when source data is available, it performs transfer learning. Additionally, when utilizing our multi-domain, utility-driven version of RDN-Boost, instance selection is employed. Notably, if we set $\alpha_S = 1$, $\alpha_T = 1$, and $w_i = 1$ for every instance i, our method becomes equivalent to the original version of RDN-Boost and loses the ability to select instances.

4.5 Instance Weighting Strategy

Weights have been widely used in machine learning to address cost-sensitive learning, where the risks associated with prediction errors vary across classes, or to mitigate class imbalance issues. In such cases, weights are often assigned at the class level. Additionally, instance-level weighting is employed in traditional models such as Support Vector Machines (SVMs) and Logistic Regression, as well as in boosting models [58] and instance-based transfer learning methods [9].

Our proposed maximization problem incorporates explicit instance weights w_i, which can be utilized to enable differentiated treatment for each instance. In particular, this work leverages these weights to mitigate the impact of imbalance across different instance groups. We propose a weighting strategy called *instance group soft balance*. This strategy allows for a smooth transition between no balancing and full balancing by adjusting a hyperparameter we call *balance factor*.

To define the instance groups, we rely on the observation that the source domain typically has a large amount of data, while the target domain data are very limited. Furthermore, relational datasets usually exhibit high imbalance ratios, with more negative instances than positive instances. Based on that, we partition the instances into four disjoint groups according to their domain and label: *Source Negative* ($\mathcal{G}_{SN}$), *Source Positive* ($\mathcal{G}_{SP}$), *Target Negative* ($\mathcal{G}_{TN}$), and *Target Positive* ($\mathcal{G}_{TP}$).

Let $G \in \{\mathcal{G}_{SN}, \mathcal{G}_{SP}, \mathcal{G}_{TN}, \mathcal{G}_{TP}\}$ denote an instance group, $N_G = |G|$ denote the number of instances in group G, and $G_{max} \in \{\mathcal{G}_{SN}, \mathcal{G}_{SP}, \mathcal{G}_{TN}, \mathcal{G}_{TP}\}$ be the group with the largest number of instances. Let also $\beta \in [0, 1]$ be the balance factor, the

hyperparameter that controls the level of balancing. For each group G, we define a group weight W_G as

$$W_G = \frac{\beta N_{G_{\max}} + (1 - \beta) N_G}{N_G}. \tag{4.21}$$

Finally, each instance i belonging to group G is assigned a weigth w_i equal to the weight of its group:

$$w_i = W_G, \text{ for each instance } i \in G. \tag{4.22}$$

When $\beta = 0$, we have $W_G = 1$ for all groups, so $w_i = 1$ for all instances. This corresponds to no balancing. In other extreme, when $\beta = 1$, we have $W_G = \frac{N_{G_{\max}}}{N_G}$, instances in smaller groups receive higher weights. This corresponds to full balancing. By adjusting β, we can smoothly transition between these two extremes, allowing for partial balancing.

We envision the possibility of designing different weighting strategies to improve model performance. For example, an alternative approach could involve weighting instances based on a metric that reflects their relevance or pertinence level to the target domain. This could be particularly useful in transfer learning scenarios where some source instances are more relevant to the target task than others. However, this is a topic for future research.

Algorithm 2 Find all valid mappings between source and target domains

function FindAllValidMappings(source vocabulary $\mathcal{V}_S$, target vocabulary $\mathcal{V}_T$, target predicate π)
    Remove π and its recursive version from $\mathcal{V}_T$
    Initialize mappings $\mathcal{M} \leftarrow \emptyset$
    for each predicate $p_S \in \mathcal{V}_S$ **do**
        if p_S is compatible with π **then**
            Partial mapping $\mu \leftarrow \{M_{p_S \mapsto \pi}\}$
            Unmapped source predicates $\mathcal{V}'_S \leftarrow \mathcal{V}_S \setminus \{p_S\}$
            Expanded mappings $\mathcal{M}' \leftarrow$ ExpandMappings($\mathcal{V}'_S$, $\mathcal{V}_T$, μ)
            $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}'$
        end if
    end for
    return $\mathcal{M}$
end function
function ExpandMappings(source vocabulary $\mathcal{V}_S$, target vocabulary $\mathcal{V}_T$, partial mapping μ)
    if $\mathcal{V}_S = \emptyset$ **then**
        return $\{\mu\}$
    end if
    Initialize mappings $\mathcal{M} \leftarrow \emptyset$
    Select the next predicate $p_S \in \mathcal{V}_S$
    Unmapped source predicates $\mathcal{V}'_S \leftarrow \mathcal{V}_S \setminus \{p_S\}$
    $\mathcal{V}_T \leftarrow \mathcal{V}_T \cup \{\varepsilon\}$ ▷ ε is the *empty predicate*
    for each predicate $p_T \in \mathcal{V}_T$ **do**
        if p_S is compatible with p_T **then**
            New partial mapping $\mu' \leftarrow \mu \cup \{M_{p_S \mapsto p_T}\}$
            if $p_T \neq \varepsilon$ **then**
                Unmapped target predicates $\mathcal{V}'_T \leftarrow \mathcal{V}_T \setminus \{p_T\}$
            else
                $\mathcal{V}'_T \leftarrow \mathcal{V}_T$
            end if
            Expanded mappings $\mathcal{M}' \leftarrow$ ExpandMappings($\mathcal{V}'_S$, $\mathcal{V}'_T$, μ')
            $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}'$
        end if
    end for
    return $\mathcal{M}$
end function

Algorithm 3 Learn $P(\pi \mid \mathbf{MB}(\pi))$ using the multi-domain, utility-driven version of RDN-Boost

Require: Initial guess ψ_0, dataset $D_\pi = D_{\pi,\mathrm{S}} \cup D_{\pi,\mathrm{T}}$, instance weights w_i, number of iterations K, number of warm-up iterations K_w, learning rates η_k, and utility hyperparameters α_S and α_T

Ensure: The CPD $P(\pi \mid \mathbf{MB}(\pi))$ for the target predicate π

for $k = 1$ to K **do**
  $D_\delta \leftarrow \emptyset$ ▷ Stage 1 begins here
  for $i \in D_\pi$ **do**
    if $k \leq K_\mathrm{w}$ **then**
      $\alpha_{\tau(i)} \leftarrow 1$
    else
      $\alpha_{\tau(i)} \leftarrow \alpha_\mathrm{S}$ if $i \in D_{\pi,\mathrm{S}}$, otherwise α_T
    end if
    $\delta_{k,i} \leftarrow w_i z_i^{1-\alpha_{\tau(i)}}(y_i - \phi_{k-1}(\mathbf{x}_i))$ ▷ See Eq. (4.20)
    $D_\delta \leftarrow D_\delta \cup \{((\mathbf{x}_i, y_i), \delta_{k,i})\}$
  end for
  $\hat{h}_k \leftarrow$ Fit a RRT from D_δ ▷ Stage 2 begins here
  $\psi_k \leftarrow \psi_{k-1} + \eta_k \hat{h}_k$
end for
return ϕ_K ▷ ϕ_K is a function of ψ_K, see Eq. (3.8)

Chapter 5

Experiments and Results

We conducted a set of experiments to evaluate the properties of our approach from different perspectives. In particular, we aim to answer the following research questions:

- **Q1.** Does UTIL-BRDN learn accurate models that outperform learning from scratch, by transferring instances from another domain?
- **Q2.** Does the proposed utility-based objective function impact transfer learning?
- **Q3.** Does UTIL-BRDN learn accurate models despite poor mapping choices?
- **Q4.** Which aspects of a mapping contribute to successful transfer to a target domain?
- **Q5.** Which mechanisms does UTIL-BRDN have to mitigate negative transfer?
- **Q6.** Does balancing across instance groups effectively enhance performance?
- **Q7.** How sensitive is UTIL-BRDN to the amount of data available for the target domain?

Question **Q1** evaluates whether it is feasible to perform transfer learning by transferring instances from a source domain, and whether our proposed method allows for learning a more accurate model compared to those learned from just a few target data. The purpose of Question **Q2** is to discuss the benefits of adopting a utility-based strategy for integrating the source and target domains in the learning process, as opposed to simpler methods. Questions **Q3** and **Q4** focus on the role of mappings, specifically examining the model's robustness to different mappings and identifying the characteristics that make a mapping effective for transfer, respectively. Since preventing negative transfer is a common challenge in transfer

learning, Question **Q5** investigates the effectiveness of UTIL-BRDN for its mitigation. Question **Q6** assesses the efficiency of our instance weighting strategy. Finally, Question **Q7** examines the performance of our models as more data become available for the target domain.

To answer these questions, we perform four types of experiments. In Section 5.2, we learn the baseline models from scratch using the original RDN-Boost algorithm. The next set of experiments, detailed in Section 5.3, evaluates UTIL-BRDN in transferring between different pairs of datasets. We also evaluate the impact of progressive exposure to more target domain data in Section 5.4. Finally, transferring from progressive noisy source data are carried out in Section 5.5.

The following hyperparameters are set using the same values of [6, 7, 39]. The number of learning iterations is $K = 10$. For each $k = 1, \ldots, K$, the corresponding RRT $\hat{h}_k$ is constrained to having up to 8 leaves, no more than 2 literals per node, and a maximum depth of 3. We also set $\psi_0 = -1.8$, meaning that the model initially assigns a probability of approximately 0.14 to all instances. Finally, the ratio of the number of negative examples to that of positive examples is 2. These values are used in all the experiments, and other hyperparameters values are evaluated with grid search.

We evaluate model performance using the area under the receiver operating characteristic curve (AUC-ROC) and the area under the precision-recall curve (AUC-PR). Each metric is averaged across multiple experimental runs, typically associated with cross-validation iterations. The CLL metric, commonly used in experiments with the original RDN-Boost, is omitted from this study. Metrics such as accuracy, precision, recall, and F1 score are not reported due to their dependence on threshold selection. Training and inference times are also excluded from the analysis, as the experiments were conducted on a cluster with heterogeneous machines, making direct time comparisons impractical.

More detailed descriptions of the experimental methodology for each set of experiments are provided in their respective sections.

5.1 Datasets

As usual in the SRL transfer learning literature, for our experiments we consider the following relational datasets:

- **IMDB.** This dataset [59] describes a movie domain and includes facts characterizing directors, actors, and movies. It is divided into five independent groups of facts, known as *mega-examples*. The goal is to predict the *workedunder* relation, which indicates if two people have worked together.

- **Cora.** This dataset consists of a collection of citations to computer science papers [60]. Each citation is described by facts about its author, venue, and title. Relations such as *sameauthor* and *sametitle* indicate when two entities are the same. It is organized into five mega-examples, and the target task is to de-duplicate venues by predicting the *samevenue* relation.

- **UW-CSE.** Introduced in [61], this dataset is used to evaluate the SRL model named Markov Logic Network (MLN). It describes an academic domain, containing facts about professors, students, courses, publications, advisories, etc. It is organized into five mega-examples, representing the areas of *artificial intelligence*, *programming languages*, *theory*, *systems*, and *graphics* from the Department of Computer Science and Engineering at the University of Washington. The target relation is *advisedby*.

- **Twitter.** This dataset [37] consists of tweets about Belgian soccer matches and is divided into two independent folds. The facts represent the *Twitter follower* network and the words of each tweet. The goal is to predict the *accounttype* relation, which categorizes an account as either a *club*, *fan*, or *news* account.

- **Yeast.** This dataset is derived from the MIPS [1] Comprehensive Yeast Genome Database [62]. It includes information on protein location, function, phenotype, and enzymes. The goal is to predict the *proteinclass* relation. The data are grouped into four independent folds.

- **NELL.** The Never-Ending Language Learner (NELL) [63] is a machine learning system that continuously reads the web, improving its ability to extract structured knowledge from hundreds of thousands of pages. We consider two domains of NELL: *Finances* and *Sports*. NELL Finances leverages different types of evidence, such as information about companies, offices, and persons, to predict the *companyeconomicsector* relation, which links a company to an economic sector. Conversely, NELL Sports uses facts from the sports domain, including data about athletes, sports, teams, and leagues, to predict the *teamplayssport* relation. Following [6, 7], we randomly split the examples of the target relation into three different folds while retaining all facts from other predicates within each fold.

Table 5.1 and Table 5.2 summarize some characteristics of these datasets. Each dataset is originally divided into a dataset-dependent number of folds, ranging from 2 to 5. In Table 5.2, the Positive Examples column indicates the number of positive

[1] Munich Information Center of Protein Sequence

Table 5.1: Dataset domain, target predicate, and recursion

Dataset	Domain	Target Predicate	Recursion
IMDB	Movies/Entertainment	*workedunder/2*	No
Cora	Academic/Publications	*samevenue/2*	No
UW-CSE	Academic/University	*advisedby/2*	No
Twitter	Social Media	accounttype	Yes
Yeast	Biology/Genomics	*proteinclass/2*	Yes
NELL Finances	Economics/Business	*companyeconomicsector/2*	Yes
NELL Sports	Sports	*teamplayssport/2*	Yes

Table 5.2: Dataset statistics

Dataset	Folds	Constants	Types	Predicates	Facts	Positive Examples
IMDB	5	297	3	6	696	382
Cora	5	2457	5	10	38336	2640
UW-CSE	5	914	9	14	2274	113
Twitter	2	273	3	3	2312	221
Yeast	4	2470	7	7	15015	369
NELL Finances	3	3340	5	10	4579	762
NELL Sports	3	4538	4	8	9236	392

instances of the target predicate, while the Facts column indicates the number of groundings of other predicates. The table also includes the total count of distinct predicates (in the Predicates column) and other relevant statistics. As indicated in Table 5.1, our experiments always allow recursion for Twitter, Yeast, NELL Finances, and NELL Sports.

5.2 Setting the Baseline: Learning From Scratch From Target Dataset

In this section, we set our baselines, called RDN-B* and RDN-B, and assess their performances with the datasets presented above. Each baseline consists in learning the CPD of the target predicate for the respective dataset from scratch using the original RDN-Boost. Learning from scratch consists in learning a model using only the target dataset. RDN-B* and RDN-B differ in the size of the target training dataset: RDN-B* is trained on a large dataset, while RDN-B is trained on a significantly smaller one. In transfer learning, the target dataset is typically small, as seen in RDN-B. However, we also report the results of RDN-B* to demonstrate the extent to which the model's performance is influenced by the amount of data available in each domain.

Table 5.3: Predictive performance of models learned from scratch from large training datasets (RDN-B*) and from small training datasets (RDN-B).

Dataset	RDN-B* (Large Data)		RDN-B (Small Data)	
	AUC-ROC	**AUC-PR**	**AUC-ROC**	**AUC-PR**
IMDB	1 ± 0	1 ± 0	1 ± 0	1 ± 0
Cora	0.76 ± 0.19	0.61 ± 0.23	0.65 ± 0.15	0.47 ± 0.15
UW-CSE	0.93 ± 0.04	0.83 ± 0.10	0.92 ± 0.04	0.78 ± 0.10
Twitter	0.989 ± 0.009	0.98 ± 0.02	0.993 ± 0.004	0.98 ± 0.01
Yeast	0.95 ± 0.04	0.92 ± 0.06	0.87 ± 0.09	0.81 ± 0.11
NELL Finances	0.74 ± 0.04	0.66 ± 0.05	0.72 ± 0.05	0.63 ± 0.07
NELL Sports	0.988 ± 0.004	0.984 ± 0.007	0.989 ± 0.006	0.97 ± 0.03

Experimental methodology. RDN-B* is evaluated using traditional cross-validation. In this method, the dataset is divided into F folds, as defined in Table 5.2, and the model is iteratively trained on $F-1$ folds while being evaluated on the fold of the respective iteration. This process is repeated for F iterations, with performance metrics such as AUC-ROC and AUC-PR being averaged over all iterations. Additionally, the standard deviation of these metrics is reported.

In contrast, RDN-B is evaluated using a variant of traditional cross-validation that is more suited to evaluate transfer learning methods. This version of cross-validation forces the model to be trained on a small training set in each iteration. Specifically, at each iteration, the model is fitted using only the fold corresponding to that iteration, and it is evaluated on the remaining $F-1$ folds. After all iterations, the average and standard deviation for AUC-ROC and AUC-PR are reported. Throughout this work, we refer to this method as *cross-validation for transfer learning.*

Limited data leads to underperforming models. Table 5.3 summarizes the performance of the baseline models on the considered datasets, noting that they neither account for transfer learning nor incorporate utilities; these aspects will be addressed in subsequent sections. In summary, the impact of restricting the size of the datasets used for learning varies across domains. A comparison between the performance of the RDN-B and RDN-B* models in Table 5.3 shows that the Cora and Yeast domains are the most affected, experiencing decreases of 0.14 and 0.11 in AUC-PR, respectively. In such cases, one can leverage transfer learning to overcome the challenges associated with limited training data. Interestingly, learning from scratch in the IMDB domain consistently achieved optimal performance in both AUC-ROC and AUC-PR, even under data availability constraints.

Our key insight for benefiting from knowledge transfer without incurring the costs of negative transfer involves parameterizing utility functions, as further discussed in the following sections.

Table 5.4: Performance of UTIL-BRDN(B).

Scenario	AUC-ROC	AUC-PR
Cora → IMDB	1 ± 0 (0%)	1 ± 0 (0%)
IMDB → Cora	0.912 ± 0.005 (+40.3%) ♢	0.78 ± 0.02 (+65.9%) ♢
IMDB → UW-CSE	0.94 ± 0.02 (+2.2%)	0.84 ± 0.06 (+7.7%)
Yeast → Twitter	0.995 ± 0.004 (+0.2%) ♢	0.987 ± 0.009 (+0.7%)
Twitter → Yeast	0.996 ± 0.002 (+14.5%)	0.990 ± 0.007 (+22.2%) ♢
NELL S → NELL F	0.79 ± 0.01 (+9.7%)	0.72 ± 0.01 (+14.3%)
NELL F → NELL S	0.997 ± 0.002 (+0.8%)	0.992 ± 0.004 (+2.3%)

5.3 Transferring Knowledge Across Domains

The next set of experiments evaluates UTIL-BRDN.

Experimental methodology. Following previous work [6, 7], the transfer scenarios are: (1) IMDB → Cora, (2) Cora → IMDB, (3) Twitter → Yeast, (4) Yeast → Twitter, (5) IMDB → UW-CSE, (6) NELL Finances → NELL Sports, and (7) NELL Sports → NELL Finances. Grid search is performed with the following hyperparameter choices: $\alpha_{\mathrm{S}} \in \{0, 0.3, 0.6, 1, 1.3\}$, $\alpha_{\mathrm{T}} \in \{0, 0.3, 0.6, 1, 1.3\}$, $K_{\mathrm{w}} \in \{1, 3, 5, 7\}$, and $\beta = 0$. We also select five mappings uniformly at random from the set of legal mappings, following the definition given in Section 4.3. In total, each pair of domains is evaluated in 500 different settings, corresponding to all possible combinations of α_{T}, α_{S}, K_{W}, and mappings. Out of these 500 settings, we refer to the model associated with the setting that led to the best average AUC-PR as UTIL-BRDN(B). Similarly, UTIL-BRDN(W) refers to the model learned with the setting associated with the worst average AUC-PR.

Each setting in the grid search is evaluated under cross-validation for transfer learning. For each cross-validation iteration, the entire source dataset is first mapped using the considered mapping. Then, it is merged with the training fold of the target dataset for that iteration. Next, the merged dataset is used to fit a model, which finally is evaluated on the test folds. Thus, the metrics are averaged and reported with their standard deviation.

Results. Tables 5.4 and 5.5 report the performances of UTIL-BRDN(B) and UTIL-BRDN(W), respectively, for each transfer scenario. The hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) are reported in Appendix A. The gains reported in parentheses in the AUC-ROC and AUC-PR columns in Tables 5.4 and 5.5 are relative to RDN-B. Additionally, we indicate statistical significance according to a paired t-test at the 0.05 significance level by the symbol ♢.

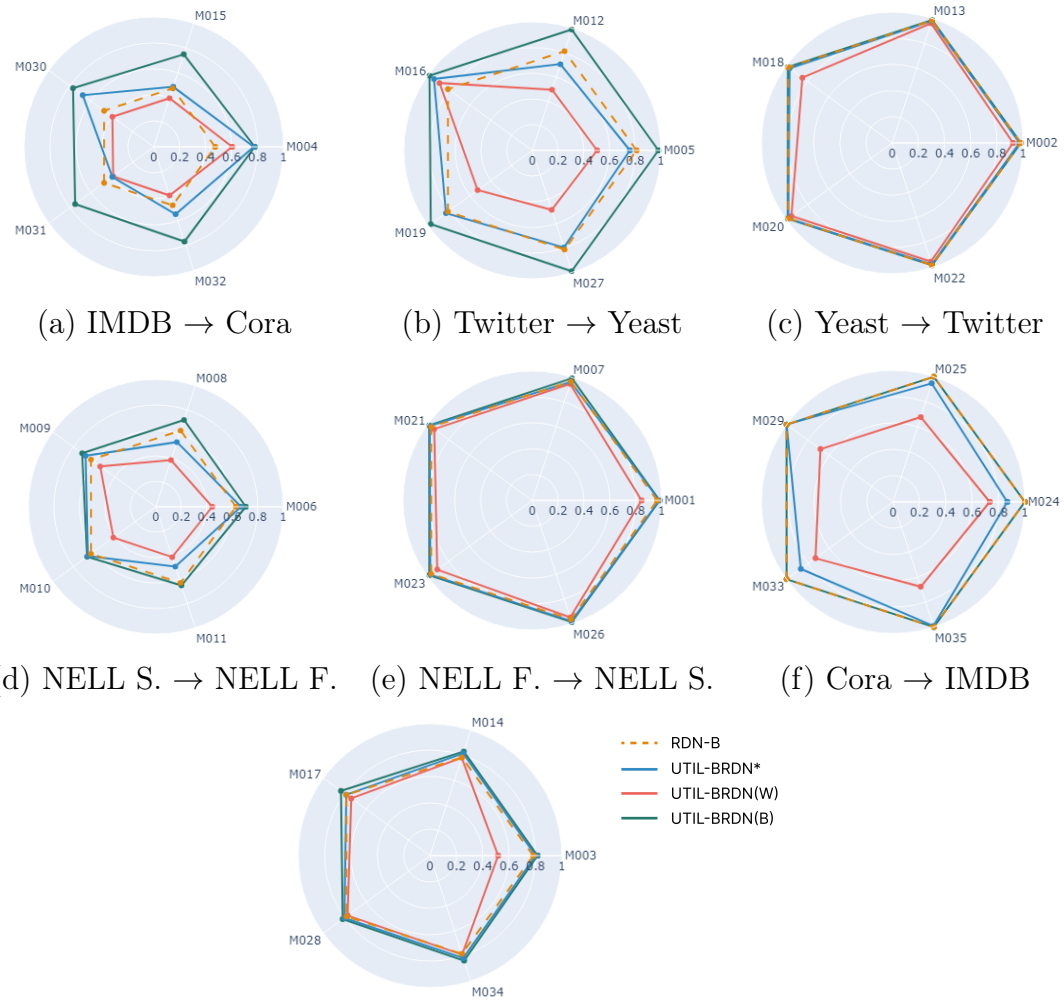
Table 5.5: Performance of UTIL-BRDN(W).

Scenario	AUC-ROC	AUC-PR
Cora → IMDB	0.78 ± 0.26 (-22%)	0.68 ± 0.33 (-32%)
IMDB → Cora	0.57 ± 0.01 (-12.3%)	0.39 ± 0.01 (-17%)
IMDB → UW-CSE	0.67 ± 0.23 (-27.2%)	0.52 ± 0.25 (-33.3%)
Yeast → Twitter	0.96 ± 0.02 (-3.3%)	0.85 ± 0.12 (-13.3%)
Twitter → Yeast	0.56 ± 0.01 (-35.6%) $\diamondsuit$	0.487 ± 0.009 (-39.9%) $\diamondsuit$
NELL S → NELL F	0.531 ± 0.005 (-26.2%) $\diamondsuit$	0.385 ± 0.008 (-38.9%) $\diamondsuit$
NELL F → NELL S	0.93 ± 0.03 (-6%)	0.85 ± 0.06 (-12.4%)

Effectiveness of transfer learning with UTIL-BRDN. Next, our focus is to answer question **Q1**, which asks whether instances from another domain can be transferred to effectively learn a better model in the target domain. Table 5.4, reporting performances for UTIL-BRDN(B), indicates that UTIL-BRDN can significantly outperform RDN-B, especially for IMDB to Cora and for Twitter to Yeast, achieving an increase in AUC-PR of 0.31 and 0.18, respectively. Thus, even though Table 5.5 suggests the opposite, it must be kept in mind that UTIL-BRDN(W) refers to the worst setting evaluated. Consequently, the answer to question **Q1** is affirmative, provided the hyperparameters are appropriately tuned.

The role of utilities. Our next focus is on question **Q2**, which aims to explore the relevance of the utilities introduced in our augmented version of RDN-Boost. In our framework, instance-based transfer learning could be performed by simply preprocessing the source and target data using the first three steps in Figure 4.1, and then feeding these preprocessed data to the original RDN-Boost rather than following steps 4 and 5 (see Section 4.4.3). In this case, RDN-Boost would learn from data by maximizing CLL. This is equivalent to learning using UTIL-BRDN with $\alpha_S = 1$, $\alpha_T = 1$, and $w_i = 1$ for every instance i. The model associated with the mapping which led to the best average AUC-PR under this naive approach is referred to as UTIL-BRDN*.

Figure 5.1 compares UTIL-BRDN* against UTIL-BRDN(B), UTIL-BRDN(W), and the baseline RDN-B, noting that the former two are now redefined to be the best and the worst setting given the mapping being used. UTIL-BRDN(B) and UTIL-BRDN(W) follow the entire pipeline in Figure 4.1, i.e., they also use our augmented version of RDN-Boost introduced in Section 4.4. This version replaces the maximization of CLL used in the original RDN-Boost and introduces a multi-domain, utility-driven maximization problem. Consequently, the hyperparameters introduced by our approach provide a mechanism for instance selection. Each vertex in the radar plots of Figure 5.1 corresponds to a learned model, its angle indicates the vocabulary mapping used in that model, and its distance to the center indicates the



(a) IMDB → Cora (b) Twitter → Yeast (c) Yeast → Twitter

(d) NELL S. → NELL F. (e) NELL F. → NELL S. (f) Cora → IMDB

(g) IMDB → UW-CSE

Figure 5.1: Average AUC-PR of models under different mappings and transfer scenarios.

resulting average AUC-PR. Each polygon contains five vertices, each corresponding to one of the five random legal mappings evaluated in the grid search for each pair of domains. Mappings are identified in the plots by a short ID (e.g., M015).

Our first observation from these plots is that UTIL-BRDN(B) (green polygon) consistently outperforms or matches the performance of RDN-B (yellow polygon) and UTIL-BRDN* (blue polygon) across all pairs of domains and evaluated mappings. On the other hand, UTIL-BRDN(W) (red polygon) indicates that UTIL-BRDN can also be strongly affected by negative transfer. This further highlights the importance of an adequate tuning of the UTIL-BRDN hyperparameters. Unexpectedly, though, UTIL-BRDN* performed poorly in most of the considered settings, the main exception being the case of transference from IMDB to Cora. We now focus on scenarios IMDB → Cora (Figure 5.1(a)), Twitter → Yeast (Figure 5.1(b)), and NELL Sports → NELL Finances (Figure 5.1(d)), aiming to highlight the importance

of utilities beyond data preprocessing.

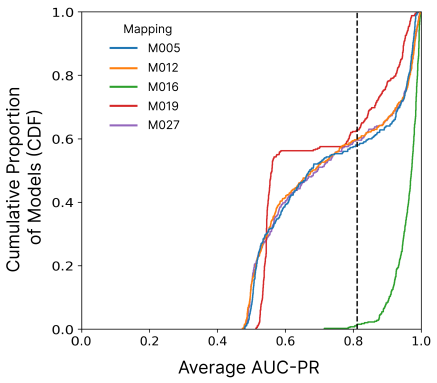
In the case of transferring from IMDB to Cora, UTIL-BRDN* outperforms RDN-B for most of the mappings, but the improvements are marginal. This indicates that some instances in the source domain are relevant to the target domain. However, the transference of poor instances prevented the model from achieving significant performance gains, as the original RDN-Boost cannot select instances. The negative impact of such poor instances is even more evident when transferring from Twitter to Yeast and from NELL Sports to NELL Finances, where UTIL-BRDN* underperforms RDN-B for most of the evaluated mappings.

In summary, answering question **Q2** is conditioned on whether hyperparameters can be effectively tuned. We have demonstrated this with the redefined UTIL-BRDN(B), so the answer is affirmative.

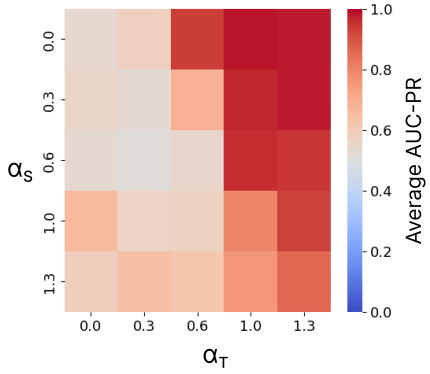
Performance and robustness of mappings. Question **Q3** focuses on how mappings impact the performance of UTIL-BRDN. To answer this question, we consider the transfer from Twitter to Yeast. Among the considered pairs of domains, this case is particularly interesting due to the existence of a mapping that stands out for its remarkable quality compared to others mappings.

Figure 5.2(a) shows the cumulative distribution function (CDF) of models (y-axis) concerning AUC-PR (x-axis). Each curve corresponds to a different mapping and encompasses all models learned with this mapping during the grid search. Thus, for each AUC-PR value on the x-axis, the corresponding value on the y-axis is the proportion of models achieving up to that AUC-PR value. A vertical dashed line indicates the AUC-PR of RDN-B. This plot illustrates that the mapping M015 is more robust than the remaining mappings. In particular, it consistently outperforms learning from scratch for almost all the evaluated hyperparameter settings, as its CDF curve is mostly to the right of the baseline in AUC-PR metrics. In contrast, models learned with the other mappings (red, orange, blue, and purple) are challenging to tune, with almost 60% of the evaluated settings resulting in models worse than the baseline. Despite this, the remaining 40% models learned for each of these mappings indicate that UTIL-BRDN can extract useful patterns even with low-quality mappings, thereby answering question **Q3** affirmatively. This ability to learn efficient models despite poor mappings represents a significant contribution of our work.

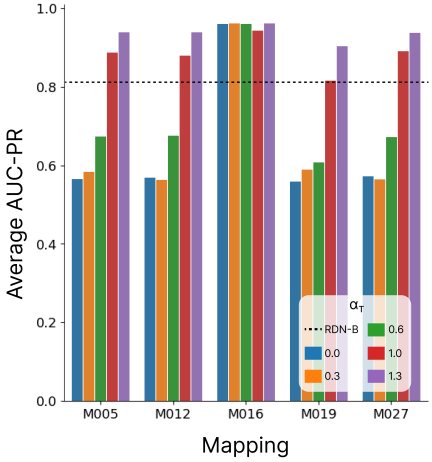
Additionally, the transfer from IMDB to Cora also suggests a high-quality and robust mapping, which contrasts with the other mappings, whose performance strongly depends on hyperparameter tuning. On the other hand, all mappings considered in the transfer from NELL Finances to NELL Sports follow a pattern which is similar to the red, orange, blue, and purple mappings in Figure 5.2(a). This likely hap-



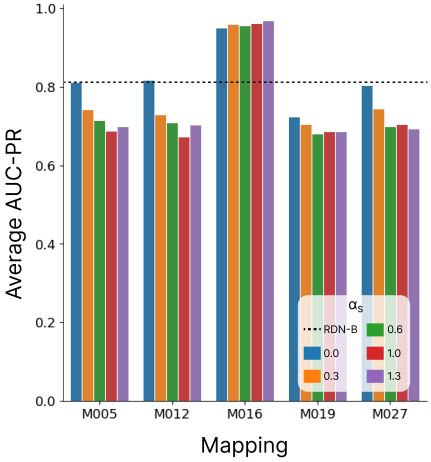
(a) CDF of model performance



(b) Impact of α_S and α_T on mapping $M027$



(c) Impact of α_T on the model



(d) Impact of α_S on the model

Figure 5.2: Various figures showing the impact of different mappings and hyperparameters on model performance when transferring from Twitter to Yeast.

pened because the random mapping selection method probably ended up selecting only suboptimal mappings. These results motivate the need for mechanisms to automatically find good mappings, which as discussed in Section 2.3 has been a central focus of previous work.

Handling negative transfer. Given the above findings, the question of how to optimally tune the UTIL-BRDN hyperparameters to achieve maximum transfer efficiency remains. This is related to question **Q5**, which addresses how to mitigate negative transfer. According to Figure 5.2(c), which shows the average performance across all transfer experiments from Twitter to Yeast given a mapping and a value for α_T, the model almost always benefits from higher values of α_T, regardless of the mapping. Similarly, Figure 5.2(d) suggests that setting α_S to a lower value is preferable in such scenario. In the specific case of transferring from Twitter to Yeast, the model is seen to be more sensitive to α_T than to α_S, as variations in α_T lead to more pronounced changes in performance.

The above behavior aligns with the heuristic introduced in TrAdaBoost [9] and discussed in Section 4.4.2, as it posits that incorrectly classified instances should be penalized by reducing their weights if they belong to the source domain, whereas misclassified instances from the target domain should be given more emphasis by increasing their weights. That is precisely how low values of α_S and high values of α_T work. While the plots in Figures 5.2(c) and 5.2(d) separately analyze the impact of α_S and α_T, the overall model performance is ultimately influenced by the interplay between these hyperparameters and the chosen mapping. We provide an example of this in Figure 5.2(b), where the value of both α_S and α_T are varied while their combined effect on AUC-PR is shown for a low-quality mapping $M027$. Unlike what is seen to happen in the case of mapping $M027$, for a robust mapping such as $M016$ the choice of hyperparameters does not significantly impact performance. Interestingly, not all pairs of domains exhibit the same pattern. For the transfer from IMDB to Cora, for example, we observed an opposite trend regarding the impact of α_S and α_T. Counterintuitively, the model performs better on average with higher values of α_S and lower values of α_T.

Based on these discussions, we also answer question **Q5** affirmatively. More specifically, the values of α_S and α_T can effectively mitigate negative transfer. Similarly, we posit that instance weights can also be leveraged for this same purpose. An evidence for this is given in Section 5.5.

5.4 Tracking the Gains of Continuously Accumulating Target Data

For each pair of domains, we perform our next experiments with RDN-B, and with the versions of UTIL-BRDN(B) and UTIL-BRDN(W) defined in Section 5.3. The goal is to evaluate how the availability of target data affects model robustness and performance, and thereby to answer question **Q7**.

Experimental methodology. This time, we adopt an evaluation methodology that extends traditional cross-validation to simulate learning from datasets with different sizes. This is achieved by adding an inner loop in the traditional cross-validation process to simulate a progressive increase in the size of the training set. At each iteration of the outer loop (which corresponds to the traditional method), the training set is shuffled and then split into five new folds. The inner loop iterates over these folds, redefining the training set and learning a new model from it, and then evaluating the resulting model on the test set associated with the outer loop's current iteration. At each iteration of the inner loop, the training set is the union

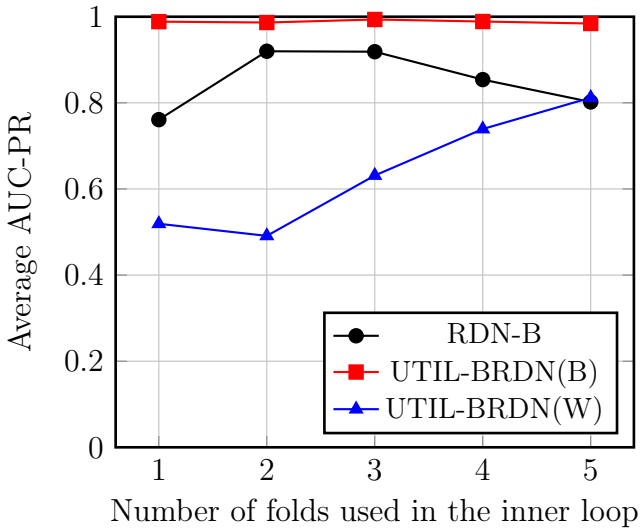


Figure 5.3: Learning curves for the average AUC-PR obtained from Twitter → Yeast.

of all folds used so far in the same iteration of the outer loop.

Benefits of large datasets. Figure 5.3 shows the learning curves for the three models considered in the transfer from Twitter to Yeast. The average AUC-PR is given on the y-axis, the number of folds used in the inner loop on the x-axis. This figure shows that UTIL-BRDN(B) is not impacted by the size of the target dataset. On the hand, UTIL-BRDN(W) is nearly always affected positively by the increasing amount of data, but this is never sufficient to make it significantly outperform RDN-B. For this specific transfer scenario, RDN-B initially benefits from the increase in the size of the training data, but interestingly this ceases to hold as the number of added folds continue to increase. As a consequence, RDN-B never outperforms UTIL-BRDN(B). Similar patterns are observed in the remaining transfer scenarios.

As for question **Q7**, we conclude that, when UTIL-BRDN is poorly parameterized, the size of the target training set is only useful to prevent the learned model from being worse than learning from scratch.

5.5 Transferring from a Noisy Source: Visualizing Mapping Quality

Success in transfer learning approaches is closely related to the selection of a source domain that is similar to the target domain [40]. In the context of transfer learning in relational domains, different mappings applied to the same source domain can produce different mapped source datasets with varying degrees of similarity to the target domain. Consequently, the mapping selection problem can be reduced to the

problem of selecting a source dataset, and techniques such as the one introduced by LUCA *et al.* [40] could be utilized to find the most suitable mapped source dataset for transfer to the considered target domain.

In this final set of experiments, we provide empirical evidence demonstrating that the level of similarity between the mapped source dataset and the target dataset is decisive when the goal is to learn an accurate model using UTIL-BRDN. Furthermore, we also evaluate our instance weighting strategy to mitigate imbalance among instance groups.

Experimental methodology. In a typical transfer learning scenario, the source domain and its corresponding mapping are explicitly chosen, which directly impacts the relevance of the source data for the transfer process. However, quantifying this relevance by simply comparing the data, without training a model, is challenging. In our subsequent experiments, we assess the quality of the source data without preselecting a specific source domain or mapping. Our strategy involves generating a source data that is highly similar to the target data and then progressively perturbing the source to gradually reduce the similarity. Our hypothesis is that highly similar source data implicitly reflect an appropriate choice of source domain and mapping, whereas low similarity indicates a poor choice.

Figure 5.4 illustrates the methodology adopted for these experiments. Broadly speaking, it simulates transfer learning from source domains with varying levels of similarity to the target domain, acting as a proxy for differences due to alternative source domains and mappings. To accomplish this, the simulation starts by leveraging high-quality source data (filled red dataset icon) to complement the target data (filled green dataset icon), both sampled from the same dataset (blue dataset icon), and systematically introduces controlled noise to the source aiming to decrease their similarity.

Then, the noisy version of the source (outlined red dataset icon) is combined with the training split of the target data to fit a model using the UTIL-BRDN pipeline depicted in Figure 4.1, which is further evaluated on the test split of the target data. The training and test sets of the target data are determined using the same variant of cross-validation for transfer learning used in the previous transfer experiments, in which each iteration involves selecting one of the F folds as the training set and the remaining as the test set. All cross-validation iterations see the same noisy version of the source data. This process is illustrated in the shadowed box at the bottom of Figure 5.4. Further details on the process of sampling highly similar source and target datasets and adding noise to the source data are provided in Appendix B.

We perform grid search across the following sets of values: $\alpha_S \in \{0, 0.3, 0.6, 1, 1.3\}$, $\alpha_T \in \{0, 0.3, 0.6, 1, 1.3\}$, $K_\alpha = 1$, $\beta \in \{0, 0.5, 1\}$, and $\rho \in$

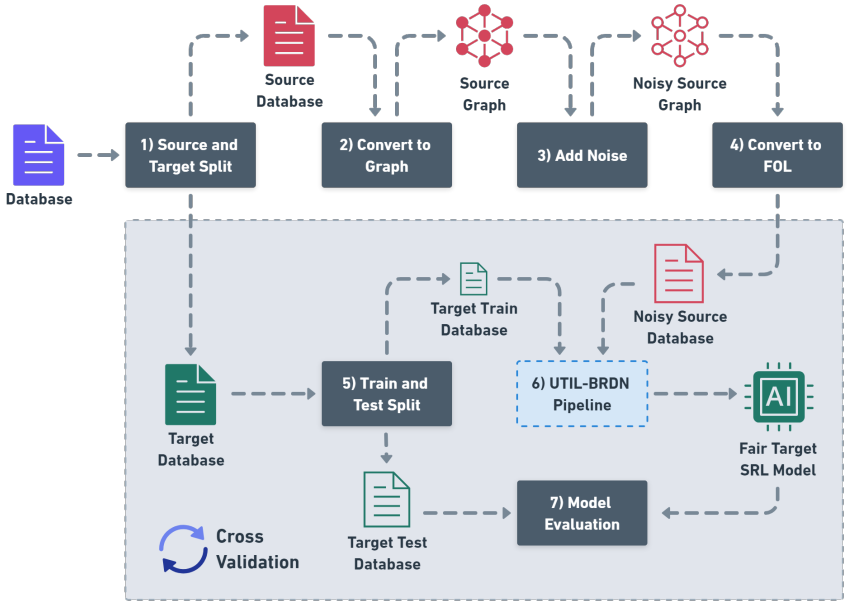


Figure 5.4: Experimental methodology to evaluate the robustness of UTIL-BRDN when learning from noisy source domains.

$\{1e^{-5} \cdot 2^i \mid i \in [0, 14]\}$. Here, ρ represents the noise strength used to decrease the similarity of the source data to the target domain (see Step 3 in Appendix B). The hyperparameters α_S, α_T, K_α, and β are associated with UTIL-BRDN, with β specifically being the balance factor for our instance group soft balance weighting strategy, introduced in Section 4.5.

This experiment does not involve sampling mappings since the source and target data are derived from the same domain. We conduct the experiments with Yeast, NELL Sports, and NELL Finances as initial domains. For each hyperparameter setting, we repeat the entire experimental pipeline, as outlined in Figure 5.4, four times using different random seeds to account for randomness in the dataset splitting when generating source and target data, and in noise addition.

Source-target similarity is key to UTIL-BRDN performance. Figures 5.5(a) and 5.5(b) show the learning curves of average AUC-PR obtained from the transfer simulation with noisy source domains under complete imbalance ($\beta = 0$) using NELL Sports. These figures illustrate how the performance varies as a function of the noise strength ρ for different values of α_S and α_T, respectively. The model suffers a performance downgrade, regardless of its parameterization, as the noise level increases. This occurs because the source data progressively becomes less similar to the target data, making it difficult to extract useful patterns for transfer.

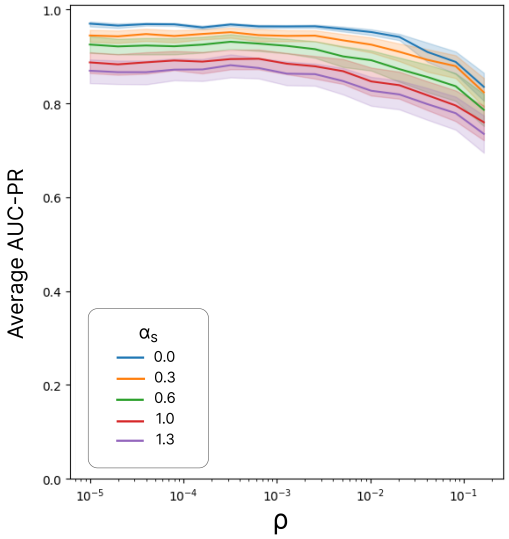
In addition to this behavior, our experiments also indicate that some domains are significantly less tolerant to dissimilarities with the source domain. This is particularly evident in the case of NELL Finances, as shown in Figure 5.5(d), which

presents the learning curves with respect to noise strength under complete imbalance ($\beta = 0$) for different values of α_S. The noise begins to affect NELL Finances much earlier than NELL Sports. We found it unnecessary to include the learning curves for NELL Finances separated by α_T, as it is very similar to Figure 5.5(d). Most of the next discussions applicable to NELL Sports also extend to NELL Finances, except for the rate at which its performance decreases due to increased noise.

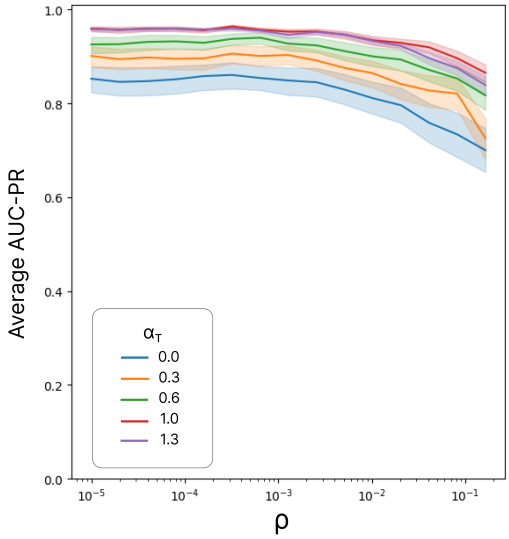
As for Question **Q4**, we empirically demonstrated that the choice of source domain and mapping, represented by datasets with varying noise levels in our simulations, directly impacts model performance because it determines the relevance of the source data to the target domain. Successful transfers are typically achieved when the mapped source data are similar to the target data, as indicated by low noise levels in our experiments. In contrast, when the source data become highly dissimilar to the target data, corresponding to higher noise levels, the model's performance degrades. The intensity of this effect varies across different target domains.

Benefits of balancing instance groups. Our previous experiments demonstrated that negative transfer can be mitigated by appropriately tuning α_{S} and α_{T}. However, Figures 5.5(a), 5.5(b), and 5.5(d) indicate that these improvements are conditioned by a given noise level, with model performance being negatively impacted on average as noise levels increase, regardless of tuning. In extreme cases of transferring from highly noisy sources, α_{S} and α_{T} may lose their effectiveness. This is evident from the strong overlap of curves for different α_{S} values in the high-noise region of Figure 5.5(d). In such cases, where tuning α_{S} and α_{T} fails to improve the model, exploring alternative solutions becomes crucial. We believe that instance weights could provide a viable solution in many of these scenarios.

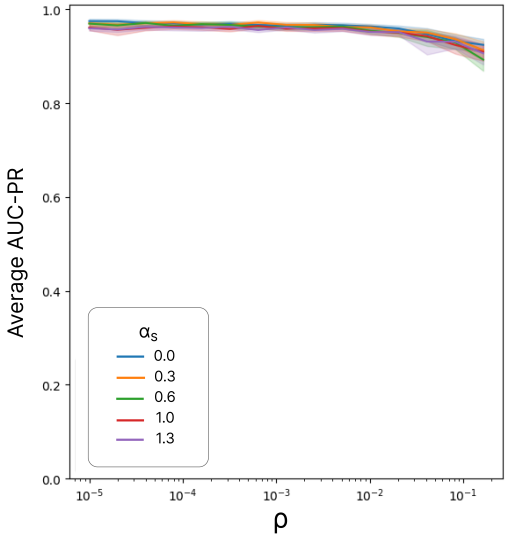
In fact, Figure 5.5(c), which presents learning curves for the case of intermediate balancing ($\beta = 0.5$) in the experiment with NELL Sports for different values of α_{S}, suggests that instance weights can further improve the performance of UTILBRDN. Compared to the fully imbalanced scenario in Figure 5.5(a), the model demonstrates increased robustness to noise under our instance weighting strategy. However, we did not find evidence that this strategy contributes positively to the model in simulations with NELL Finances and Yeast, leading us to conclude that Question **Q6** can be answered affirmatively for some domains but not universally.



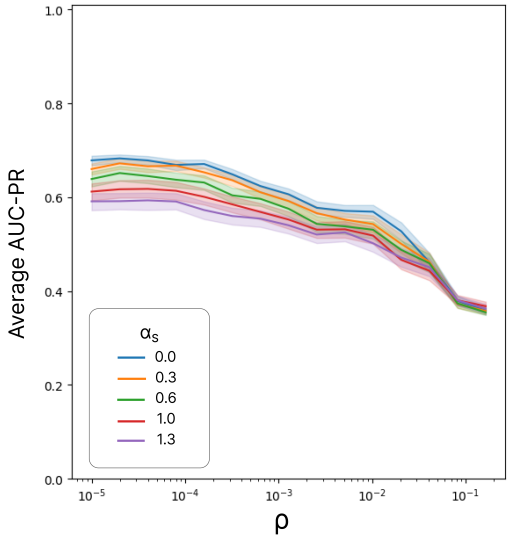
(a) NELL Sports: $\beta = 0$, grouped by α_S



(b) NELL Sports: $\beta = 0$, grouped by α_T



(c) NELL Sports: $\beta = 0.5$, grouped by α_S



(d) NELL Finances: $\beta = 0$, grouped by α_S

Figure 5.5: Learning curves of performance with respect to noise strength and α-values for different domains.

Chapter 6

Conclusion

This work introduced UTIL-BRDN, an instance-based transfer learning approach specifically designed for statistical relational learning. UTIL-BRDN leverages utilities to enhance the performance of models resulting from the transfer. It operates within a two-stage framework: first, it preprocesses data by mapping instances from the source domain and integrating them with instances from the target domain; then, it feeds this combined data to an SRL model augmented to incorporate the utilities from both domains. While we focus on RDN-Boost, our approach is general enough to accommodate alternative SRL models such as MLNs. To our knowledge, this is the first work in the SRL transfer learning literature to introduce the combined use of instance-based transfer learning and utilities. Previous works typically concentrate on model-based transfer learning and do not account for utilities. We also proposed an instance weighting strategy that implements a flexible mechanism to mitigate the impact of class and domain imbalance.

We extensively evaluated UTIL-BRDN through a series of transfer experiments using seven widely used real-world relational datasets. Different transfer settings were tested, and the results were compared against both learning from scratch using the original RDN-Boost and a variant of UTIL-BRDN without utilities. Additionally, we described several insights regarding how predictive performance relates to utilities, the chosen mapping, the amount of data available in the target domain, and the instance weights assigned under our weighting strategy.

Our results demonstrate that UTIL-BRDN effectively transfers knowledge by leveraging instances from the source domain. Specifically, an appropriate parameterization of UTIL-BRDN led to significant improvements in model performance, as observed in the transfers from IMDB to Cora (an improvement of 0.31 in AUC-PR) and from Twitter to Yeast (an improvement of 0.18 in AUC-PR). Furthermore, a proper tuning of how utilities are weighted makes UTIL-BRDN more robust to poor mappings. This suggests that it has an intrinsic mechanism to mitigate negative transfer and that properly tuning its hyperparameters acts as an instance selection

process. This explains why UTIL-BRDN with well-tuned hyperparameters never performed worse than the baseline in our experiments.

This work opens up a number of directions for further investigation. First, the methodology can be adapted to other SRL models, such as MLNs. Second, we envision the design of new search algorithms to find good mappings, or the adaptation of existing heuristic approaches, such as the ranked-based version of the mapping search algorithm that uses word embeddings [7]. Third, exploring alternative instance weighting strategies beyond the one proposed in this work to alleviate the imbalance between different groups of instances seems promising. Additionally, we envision extending the proposed approach to multiple source domains and incorporating utility-based optimization into traditional, propositional transfer learning methods, particularly for instance selection.

References

[1] GETOOR, L., TASKAR, B. *Introduction to statistical relational learning*. Cambridge, MA, MIT press, 2007.

[2] RAEDT, L. D., KERSTING, K. "Statistical Relational Learning". In: Sammut, C., Webb, G. I. (Eds.), *Encyclopedia of Machine Learning*, Springer, pp. 916–924, Cham, 2010. doi: 10.1007/978-0-387-30164-8_786. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_786>.

[3] RAEDT, L. D., KERSTING, K., NATARAJAN, S., et al. "Statistical relational artificial intelligence: Logic, probability, and computation", *Synthesis lectures on artificial intelligence and machine learning*, v. 10, n. 2, pp. 1–189, 2016.

[4] PAN, S. J., YANG, Q. "A survey on transfer learning", *IEEE Transactions on knowledge and data engineering*, v. 22, n. 10, pp. 1345–1359, 2009.

[5] MIHALKOVA, L., HUYNH, T., MOONEY, R. J. "Mapping and revising markov logic networks for transfer learning". In: *Aaai*, v. 7, pp. 608–614, 2007.

[6] AZEVEDO SANTOS, R., PAES, A., ZAVERUCHA, G. "Transfer learning by mapping and revising boosted relational dependency networks", *Machine Learning*, v. 109, pp. 1435–1463, 2020.

[7] LUCA, T., PAES, A., ZAVERUCHA, G. "Word embeddings-based transfer learning for boosted relational dependency networks", *Machine Learning*, pp. 1–34, 2023.

[8] LIN, D., AN, X., ZHANG, J. "Double-bootstrapping source data selection for instance-based transfer learning", *Pattern Recognition Letters*, v. 34, n. 11, pp. 1279–1285, 2013.

[9] DAI, W., YANG, Q., XUE, G.-R., et al. "Boosting for transfer learning". In: *Proceedings of the 24th international conference on Machine learning*, pp. 193–200, 2007.

[10] KELLY, F. "Charging and rate control for elastic traffic", *European transactions on Telecommunications*, v. 8, n. 1, pp. 33–37, 1997.

[11] BALSEIRO, S., LU, H., MIRROKNI, V. "Regularized Online Allocation Problems: Fairness and Beyond". In: Meila, M., Zhang, T. (Eds.), *Proceedings of the 38th International Conference on Machine Learning*, v. 139, *Proceedings of Machine Learning Research*, pp. 630–639, virtual conference, 18–24 Jul 2021. PMLR. Disponível em: <`https://proceedings.mlr.press/v139/balseiro21a.html`>.

[12] ZHANG, G., MALEKMOHAMMADI, S., CHEN, X., et al. "Proportional Fairness in Federated Learning", *Transactions on Machine Learning Research – arXiv preprint arXiv:2202.01666*, 2022.

[13] FRIEDMAN, C., SANDOW, S. *Utility-based learning from data.* Chapman and Hall/CRC, 2016.

[14] JU, P., GHOSH, A., SHROFF, N. B. "Achieving Fairness in Multi-Agent Markov Decision Processes Using Reinforcement Learning", *arXiv preprint arXiv:2306.00324*, 2023.

[15] MOSHAHEDI, N., KATTAN, L. "Alpha-fair large-scale urban network control: A perimeter control based on a macroscopic fundamental diagram", *Transportation Research Part C: Emerging Technologies*, v. 146, pp. 103961, 2023.

[16] ALLYBOKUS, Z. *Real-time scalable algorithms for alpha-fair resource allocation in software defined networks.* Tese de Doutorado, Université Côte d'Azur, 2019.

[17] PEROVŠEK, M., VAVPETIČ, A., KRANJC, J., et al. "Wordification: Propositionalization by unfolding relational data into bags of words", *Expert Systems with Applications*, v. 42, n. 17-18, pp. 6442–6456, 2015.

[18] ŽELEZNỲ, F., LAVRAČ, N. "Propositionalization-based relational subgroup discovery with RSD", *Machine Learning*, v. 62, pp. 33–63, 2006.

[19] FRANÇA, M. V., ZAVERUCHA, G., D'AVILA GARCEZ, A. S. "Fast relational learning using bottom clause propositionalization with artificial neural networks", *Machine learning*, v. 94, pp. 81–104, 2014.

[20] BORDES, A., USUNIER, N., GARCIA-DURAN, A., et al. "Translating embeddings for modeling multi-relational data", *Advances in neural information processing systems*, v. 26, 2013.

[21] WANG, Z., ZHANG, J., FENG, J., et al. "Knowledge graph embedding by translating on hyperplanes". In: *Proceedings of the AAAI conference on artificial intelligence*, v. 28, 2014.

[22] LIN, Y., LIU, Z., SUN, M., et al. "Learning entity and relation embeddings for knowledge graph completion". In: *Proceedings of the AAAI conference on artificial intelligence*, v. 29, 2015.

[23] WU, Z., PAN, S., CHEN, F., et al. "A comprehensive survey on graph neural networks", *IEEE transactions on neural networks and learning systems*, v. 32, n. 1, pp. 4–24, 2020.

[24] ZHANG, S., TONG, H., XU, J., et al. "Graph convolutional networks: a comprehensive review", *Computational Social Networks*, v. 6, n. 1, pp. 1–23, 2019.

[25] VELIČKOVIĆ, P., CUCURULL, G., CASANOVA, A., et al. "Graph attention networks", *arXiv preprint arXiv:1710.10903*, 2017.

[26] CROPPER, A., DUMANČIĆ, S. "Inductive logic programming at 30: a new introduction", *Journal of Artificial Intelligence Research*, v. 74, pp. 765–850, 2022.

[27] QUINLAN, J. R., CAMERON-JONES, R. M. "FOIL: A midterm report". In: *Machine Learning: ECML-93: European Conference on Machine Learning Vienna, Austria, April 5–7, 1993 Proceedings 6*, pp. 1–20. Springer, 1993.

[28] MUGGLETON, S. "Inverse entailment and Progol", *New generation computing*, v. 13, pp. 245–286, 1995.

[29] BLOCKEEL, H., DE RAEDT, L. "Top-down induction of first-order logical decision trees", *Artificial intelligence*, v. 101, n. 1-2, pp. 285–297, 1998.

[30] FRIEDMAN, N., GETOOR, L., KOLLER, D., et al. "Learning probabilistic relational models". In: *IJCAI*, v. 99, pp. 1300–1309, 1999.

[31] NEVILLE, J., JENSEN, D. "Relational dependency networks." *Journal of Machine Learning Research*, v. 8, n. 3, 2007.

[32] MIHALKOVA, L. S. *Learning with Markov logic networks: transfer learning, structure learning, and an application to Web query disambiguation*. Texas, Austin, The University of Texas at Austin, 2009.

[33] NATARAJAN, S., KHOT, T., KERSTING, K., et al. "Gradient-based boosting for statistical relational learning: The relational dependency network case", *Machine Learning*, v. 86, pp. 25–56, 2012.

[34] JIANG, J., ZHAI, C. "Instance weighting for domain adaptation in NLP". ACL, 2007.

[35] MIHALKOVA, L., MOONEY, R. J. "Transfer Learning from Minimal Target Data by Mapping across Relational Domains." In: *IJCAI*, v. 9, pp. 1163–1168, 2009.

[36] DAVIS, J., DOMINGOS, P. "Deep transfer via second-order markov logic". In: *Proceedings of the 26th annual international conference on machine learning*, pp. 217–224, 2009.

[37] VAN HAAREN, J., KOLOBOV, A., DAVIS, J. "TODTLER: Two-order-deep transfer learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 29, 2015.

[38] KUMARASWAMY, R., ODOM, P., KERSTING, K., et al. "Transfer learning via relational type matching". In: *2015 IEEE International Conference on data mining*, pp. 811–816. IEEE, 2015.

[39] DE FIGUEIREDO, L. F., PAES, A., ZAVERUCHA, G. "Transfer learning for boosted relational dependency networks through genetic algorithm". In: *International Conference on Inductive Logic Programming*, pp. 125–139. Springer, 2021.

[40] LUCA, T., PAES, A., ZAVERUCHA, G. "Select First, Transfer Later: Choosing Proper Datasets for Statistical Relational Transfer Learning". In: *International Conference on Inductive Logic Programming*, pp. 62–76. Springer, 2023.

[41] RICHARDS, B. L., MOONEY, R. J. "Automated refinement of first-order horn-clause domain theories", *Machine Learning*, v. 19, pp. 95–131, 1995.

[42] YANG, Q., ZHANG, Y., DAI, W., et al. *Transfer Learning*. Cambridge University Press, 2020.

[43] ZHUANG, F., QI, Z., DUAN, K., et al. "A comprehensive survey on transfer learning", *Proceedings of the IEEE*, v. 109, n. 1, pp. 43–76, 2020.

[44] HOSNA, A., MERRY, E., GYALMO, J., et al. "Transfer learning: a friendly introduction", *Journal of Big Data*, v. 9, n. 1, pp. 102, 2022.

[45] WEISS, K., KHOSHGOFTAAR, T. M., WANG, D. "A survey of transfer learning", *Journal of Big data*, v. 3, pp. 1–40, 2016.

[46] WANG, Z., DAI, Z., PÓCZOS, B., et al. "Characterizing and avoiding negative transfer". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11293–11302, 2019.

[47] SHIMODAIRA, H. "Improving predictive inference under covariate shift by weighting the log-likelihood function", *Journal of statistical planning and inference*, v. 90, n. 2, pp. 227–244, 2000.

[48] QUINLAN, J. R. *C4.5: programs for machine learning*. San Francisco, CA, United States, Elsevier, 2014.

[49] HECKERMAN, D., CHICKERING, D. M., MEEK, C., et al. "Dependency networks for inference, collaborative filtering, and data visualization", *Journal of Machine Learning Research*, v. 1, n. Oct, pp. 49–75, 2000.

[50] NATARAJAN, S., KERSTING, K., KHOT, T., et al. *Boosted statistical relational learners: From benchmarks to data-driven medicine*. Berlin, Germany, Springer, 2015.

[51] DIETTERICH, T. G., HAO, G., SCHENFELTER, A. "Gradient Tree Boosting for Training Conditional Random Fields." *Journal of Machine Learning Research*, v. 9, n. 10, 2008.

[52] SHARMA, S., GUPTA, M., GOYAL, K., et al. "Identifying Parkinson's Patients by a Functional Gradient Boosting Approach". In: *Intelligent Technologies and Parkinson's Disease: Prediction and Diagnosis*, IGI Global, pp. 288–304, 2024.

[53] NATARAJAN, S., KERSTING, K., IP, E., et al. "Early prediction of coronary artery calcification levels using machine learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 27, pp. 1557–1562, 2013.

[54] GUTMANN, B., KERSTING, K. "TildeCRF: Conditional random fields for logical sequences". In: *European Conference on Machine Learning*, pp. 174–185. Springer, 2006.

[55] KHOT, T., NATARAJAN, S., KERSTING, K., et al. "Gradient-based boosting for statistical relational learning: the Markov logic network and missing data cases", *Machine Learning*, v. 100, pp. 75–100, 2015.

[56] DEHGHAN, M., MASSOULIE, L., TOWSLEY, D., et al. "A utility optimization approach to network cache design", *IEEE/ACM Transactions on Networking*, v. 27, n. 3, pp. 1013–1027, 2019.

[57] CHEN, V. X., HOOKER, J. "A guide to formulating equity and fairness in an optimization model", *Preprint*, pp. 162–174, 2021.

[58] YING, C., QI-GUANG, M., JIA-CHEN, L., et al. "Advance and prospects of AdaBoost algorithm", *Acta Automatica Sinica*, v. 39, n. 6, pp. 745–758, 2013.

[59] MIHALKOVA, L., MOONEY, R. J. "Bottom-up learning of Markov logic network structure". In: *Proceedings of the 24th international conference on Machine learning*, pp. 625–632, 2007.

[60] SINGLA, P., DOMINGOS, P. "Discriminative training of Markov logic networks". In: *AAAI*, v. 5, pp. 868–873, 2005.

[61] RICHARDSON, M., DOMINGOS, P. "Markov logic networks", *Machine learning*, v. 62, pp. 107–136, 2006.

[62] MEWES, H.-W., FRISHMAN, D., GÜLDENER, U., et al. "MIPS: a database for genomes and protein sequences", *Nucleic acids research*, v. 30, n. 1, pp. 31–34, 2002.

[63] CARLSON, A., BETTERIDGE, J., KISIEL, B., et al. "Toward an architecture for never-ending language learning". In: *Proceedings of the AAAI conference on artificial intelligence*, v. 24, pp. 1306–1313, 2010.

[64] SANFELIU, A., FU, K.-S. "A distance measure between attributed relational graphs for pattern recognition", *IEEE transactions on systems, man, and cybernetics*, , n. 3, pp. 353–362, 1983.

[65] TORRA, V., SALAS, J. "Graph perturbation as noise graph addition: a new perspective for graph anonymization". In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26–27, 2019, Proceedings 14*, pp. 121–137. Springer, 2019.

Appendix A

Additional Experimental Results

In this appendix we provide additional experimental results on the hyperparameters found when transferring knowledge across domains in Section 5.3.

- Tables A.1 and A.2: transfer between Cora and IMDB.
- Table A.3: transfer from IMDB to UW-CSE.
- Tables A.4 and A.5: transfer between Yeast and Twitter.
- Tables A.6 and A.7: transfer between NELL Sports and NELL Finances.

For each case, we report the hyperparameters in the best (UTIL-BRDN(B)) and worst (UTIL-BRDN(W)) configurations.

Table A.1: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from Cora to IMDB.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	1
$K_{\mathbf{w}}$	1
Predicate Mapping	[sameauthor, samebib, sametitle, samevenue, author, haswordauthor, haswordtitle, haswordvenue] → null, venue → workedunder, title → genre
Term Type Mapping	class → person, venue → person, title → genre

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	0.6
$\alpha_{\mathbf{T}}$	0
$K_{\mathbf{w}}$	5
Predicate Mapping	[sameauthor, samebib, sametitle, samevenue, author, venue, haswordtitle, haswordvenue] → null, title → workedunder, haswordauthor → movie
Term Type Mapping	class → person, title → person, author → movie, word → person

Table A.2: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from IMDB to Cora.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	0.3
$\alpha_{\mathbf{T}}$	1
$K_{\mathbf{w}}$	3
Predicate Mapping	[female, actor, director, movie] → null, workedunder → samevenue, genre → haswordvenue
Term Type Mapping	person → venue, genre → word

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	0.6
$K_{\mathbf{w}}$	1
Predicate Mapping	[workedunder, female, actor, director, movie] → null, genre → samevenue
Term Type Mapping	person → venue, genre → venue

Table A.3: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from IMDB to UW-CSE.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	1
$\alpha_{\mathbf{T}}$	0.6
$K_{\mathbf{w}}$	1
Predicate Mapping	[workedunder, actor] → null, movie → advisedby, female → student, director → professor, genre → yearsinprogram
Term Type Mapping	movie → person, person → person, genre → year

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	0
$K_{\mathbf{w}}$	1
Predicate Mapping	[female, actor, director] → null, workedunder → advisedby, movie → tempadvisedby, genre → hasposition
Term Type Mapping	person → person, movie → person, genre → faculty

Table A.4: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from Yeast to Twitter.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	1.3
$K_{\mathbf{w}}$	1
Predicate Mapping	[interaction, proteinclass, complex, phenotype] → null, recursion_function → recursion_accounttype, function → accounttype, location → tweets, enzyme → follows
Term Type Mapping	protein → account, fun → type, loc → word, enz → account

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	1.3
$\alpha_{\mathbf{T}}$	0
$K_{\mathbf{w}}$	1
Predicate Mapping	[interaction, enzyme, complex, phenotype] → null, recursion_location → recursion_accounttype, location → accounttype, proteinclass → follows, function → tweets
Term Type Mapping	protein → account, loc → type, class → account, fun → word

Table A.5: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from Twitter to Yeast.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	0.6
$K_{\mathbf{w}}$	1
Predicate Mapping	accounttype → proteinclass, recursion_accounttype → recursion_proteinclass, tweets → phenotype, follows → interaction
Term Type Mapping	account → protein, type → class, word → phe

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	0.3
$\alpha_{\mathbf{T}}$	0
$K_{\mathbf{w}}$	1
Predicate Mapping	tweets → enzyme, follows → interaction, accounttype → proteinclass, recursion_accounttype → recursion_proteinclass
Term Type Mapping	account → protein, type → class, word → enz

Table A.6: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from NELL Sports to NELL Finances.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	1.3
$K_{\mathbf{w}}$	7
Predicate Mapping	[athleteledsportsteam, athleteplaysinleague, teamplayssport] → null, teamplaysinleague → companyeconomicsector, recursion_teamplaysinleague → recursion_companyeconomicsector, athleteplaysforteam → acquired, athleteplayssport → companyceo, teamalsoknownas → bankboughtbank, teamplaysagainstteam → companyalsoknownas
Term Type Mapping	sportsteam → company, sportsleague → sector, athlete → company, sport → person

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	0
$K_{\mathbf{w}}$	3
Predicate Mapping	[athleteledsportsteam, athleteplaysinleague, teamplayssport] → null, teamplaysinleague → companyeconomicsector, recursion_teamplaysinleague → recursion_companyeconomicsector, athleteplaysforteam → acquired, athleteplayssport → companyceo, teamalsoknownas → bankboughtbank, teamplaysagainstteam → companyalsoknownas
Term Type Mapping	sportsteam → company, sportsleague → sector, athlete → company, sport → person

Table A.7: Hyperparameter settings of UTIL-BRDN(B) and UTIL-BRDN(W) in the transfer from NELL Finances to NELL Sports.

UTIL-BRDN(B)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	0
$K_{\mathbf{w}}$	5
Predicate Mapping	[bankboughtbank, companyalsoknownas, acquired, companyeconomicsector, recursion_companyeconomicsector, companyceo] → null, athleteplaysinleague → teamalsoknownas, companyeconomicsector → recursion_teamplaysinleague, companyheadquartercity → teamplaysagainstteam, companygpeheadquartercity → athleteplaysforteam, acquired → teamplayssport, companyceo → athleteledsportsteam, bankchiefexecutiveofficer → athleteplayssport
Term Type Mapping	company → sportsteam, sector → athlete, person → sport, sector → sport

UTIL-BRDN(W)	
$\alpha_{\mathbf{S}}$	0
$\alpha_{\mathbf{T}}$	1.6
$K_{\mathbf{w}}$	3
Predicate Mapping	[bankboughtbank, companyalsoknownas, acquired, companyeconomicsector, recursion_companyeconomicsector, companyceo] → null, athleteplaysinleague → teamalsoknownas, companyeconomicsector → recursion_teamplaysinleague, companyheadquartercity → teamplaysagainstteam, companygpeheadquartercity → athleteplaysforteam, acquired → teamplayssport, companyceo → athleteledsportsteam, bankchiefexecutiveofficer → athleteplayssport
Term Type Mapping	company → sportsteam, sector → athlete, person → sport, sector → sport

Appendix B

Controlling Source-Target Similarity Using Systematic Noise

This section provides additional details on the first four steps of the experimental methodology adopted in the experiments of Section 5.5, as illustrated in Figure 5.4. These steps include sampling similar source and target datasets in Step 1, converting the source dataset into a graph representation in Step 2, systematically adding noise to the source dataset in Step 3, and converting the perturbed data back into a FOL knowledge base in Step 4. The resulting source data is expected to be less similar to the target dataset, with the level of dissimilarity being influenced by a hyperparameter called *noise strength*.

Step 1: Sampling Similar Source and Target Datasets. Generating similar source and target datasets involves selecting an existing dataset (represented by the blue dataset icon in Figure 5.4), such as one of those considered in this work, and then uniformly splitting at random this dataset into source and target data in a proportion of 80% and 20%, respectively. Since both the source and target data are sampled from the same dataset, they are expected to be similar.

Step 2: Converting Source Dataset to Graph Next, we discuss how the source data are converted into a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of directed edges. Let L_V and L_E denote the sets of node labels and edge labels, respectively. Node labels L_V represent entity types (e.g., *person* and *company*), while edge labels L_E represent different relation types (e.g., *parent_of* and *works_at*). Each node $v \in V$ is associated with a label $l_v \in L_V$, representing an instance of the entity type l_v, also referred to as an object. Similarly, each edge $e_{ij} = (v_i, v_j) \in E$ is associated with a label $l_{e_{ij}} \in L_E$, representing an instance of the relation type $l_{e_{ij}}$.

Typically, vertices and edges may have additional attributes to describe proper-

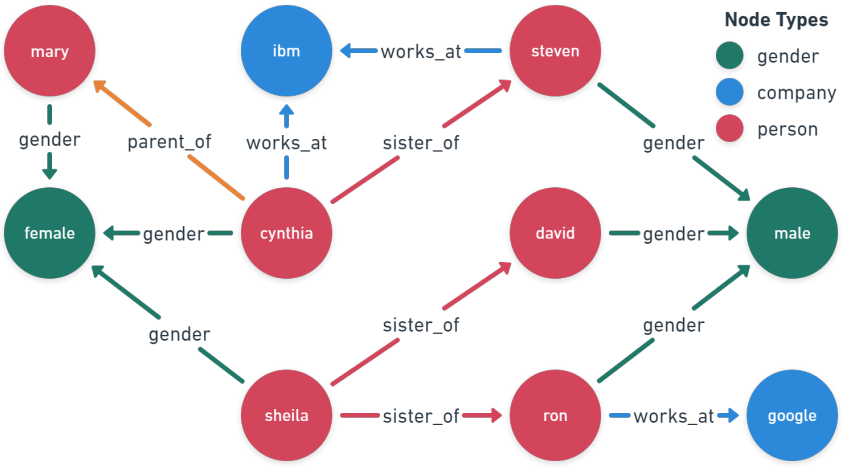


Figure B.1: A toy example of a graph database.

ties of both objects (e.g., *gender*) and relationships (e.g., *admission_date*). However, we simplify the representation by adding additional nodes to represent these attributes. For each attribute-value pair of a specific object, we create a new node that represents the attribute value and add a directed edge from the object node to the node associated with the attribute value, with the edge labeled by the attribute name. This design choice prepares the data for the noise addition algorithm presented in the next step. Figure B.1 provides an example of a dataset converted into a graph.

Algorithm 4 describes the process of converting a FOL knowledge base into a graph. The algorithm first converts all predicates to arity 2 using the procedure *ConvertPredicatesToArityTwo*. Converting examples from a binary predicate into graph elements is straightforward: each term in the predicate is mapped to a node in the graph, and an edge, labeled by the predicate name, is added between these nodes. After the predicates binarization, the procedure *ConvertFOLDatabaseToGraph* iterates over each fact and positive example, mapping them to corresponding nodes and edges in the graph.

Step 3: Adding Noise to Source Data. To motivate our method for introducing noise into the source data, we first consider how similarities between graphs can be calculated. Graph similarity can be measured using metrics such as graph edit distance [64], which defines the similarity between two graphs based on the minimum number of operations (e.g., additions, deletions, or modifications of edges or nodes) required to transform one graph into the other. This concept of graph edit distance guides our approach to systematically introducing noise by progressively editing the source graph to reduce its similarity to the target graph. Our approach is based on the graph perturbation approach proposed in [65], in which edit operations are performed by adding a randomly generated noise graph to the source graph.

Our extension of the graph perturbation method proposed in [65] involves adding noise to graphs with typed nodes and edges. Noise is represented by a random graph $G_\epsilon = (V, E_\epsilon)$, where the set of vertices is identical to the set of vertices of the original graph $G = (V, E)$, whereas the set of edges E_ϵ of the noise graph is generated at random.

For each pair of vertices in the noise graph G_ϵ, the types of edges that can be added are determined based on the types of the vertices in the pair. For example, if both vertices are of type *person*, then edges of type *sister_of* and *parent_of* are allowed according to the example dataset in Table 3.1. In such cases, one of these edge types is uniformly sampled for the respective pair of vertices and added to the set of edges E_ϵ in the noise graph with probability ρ, also called *noise strength*. In contrast, if no edge type can connect the respective pair of vertices, such as in pairs of vertices associated with the type *company*, they remain disconnected.

Next, we add the noise graph to the original to generate the perturbed version, denoted as $G' = (V, E')$. The vertices set of the perturbed graph is the same as that of the original and noise graphs, but its edges are determined using the operator defined in Table B.1 for each pair of vertices. This operator captures four types of edge edit operations on the original graph: addition, removal, type change, and preservation of the original edge state. Thus, adding the noise graph corresponds to performing a series of random edits on the original graph. The intensity of these edits is controlled by the hyperparameter ρ.

Table B.1: Definition of the graph addition operator used in graph perturbation.

Noise Graph (G_ϵ)	**Original Graph** (G)	**Perturbed Graph** ($G' = G + G_\epsilon$)	**Edit Operation**
no edge	no edge	no edge	preserve edge state
no edge	edge of type t	edge of type t	preserve edge state
edge of type t	no edge	edge of type t	add edge
edge of type t	edge of type t	no edge	remove edge
edge of type t_2	edge of type t_1, $t_1 \neq t_2$	edge of type t_2	change edge type

Step 4: Converting Perturbed Graph into a FOL Knowledge Base. After adding noise to the source data graph, the next step is to convert the perturbed graph G' back into a FOL knowledge base. Algorithm 5 outlines this process. The algorithm iterates over each edge in the graph and reconstructs the corresponding binary predicates, effectively reversing the conversion performed in Algorithm 4. The target predicate π is used to identify positive examples within the database.

Algorithm 4 Convert First-Order Logic Knowledge Base to Graph

function ConvertFOLDatabaseToGraph(FOL knowledge base $\mathcal{D}$)
    Converted knowledge base $\mathcal{D}' \leftarrow$ ConvertPredicatesToArityTwo($\mathcal{D}$)
    Background knowledge $B \leftarrow$ Extract background knowledge facts from $\mathcal{D}'$
    Positive examples $P \leftarrow$ Extract set of positive examples from $\mathcal{D}'$
    Set of all facts $F \leftarrow B \cup P$
    Set of vertices $V \leftarrow \emptyset$
    Set of edges $E \leftarrow \emptyset$
    for each fact $f \in F$ **do** Vertices
        $v_1, v_2 \leftarrow$ Extract the terms in f
        Vertex types $l_{v_1}, l_{v_2} \leftarrow$ Extract the term types in f
        Edge type $l_{(v_1,v_2)} \leftarrow$ Extract the predicate name of f
        $V \leftarrow V \cup \{v_1, v_2\}$
        $E \leftarrow E \cup \{(v_1, v_2)\}$
    end for Graph $G \leftarrow (V, E)$
    return G
end function

function ConvertPredicatesToArityTwo(FOL knowledge base $\mathcal{D}$)
    Initialize converted knowledge base $\mathcal{D}' \leftarrow \emptyset$
    for each fact $f \in \mathcal{D}$ **do**
        Predicate name $p \leftarrow$ extract predicate name from f
        Arity $n \leftarrow$ extract arity of predicate p
        if $n = 1$ **then** ▷ Unary predicate
            Extract term t and its type T from f
            Converted fact $f' \leftarrow [T]\_has\_label([t], [p])$
            $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{f'\}$
        else if $n = 2$ **then** ▷ Binary predicate
            $f' \leftarrow f$
            $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{f'\}$
        else ▷ Predicate with arity greater than 2
            Assign unique identifier $\mathcal{I}_f$ to the fact f
            for each term t and its type T in f **do**
                Converted fact $f' \leftarrow [p]\_[T]([\mathcal{I}_f], [t])$
                $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{f'\}$
            end for
        end if
    end for
    return $\mathcal{D}'$
end function

Algorithm 5 Convert Graph to First-Order Logic Knowledge Base

Require: Directed graph $G = (V, E)$ with labeled vertices and edges, target predicate π

Ensure: Database D with background knowledge facts and positive examples

1: Initialize converted knowledge base D
2: **for** each edge $e = (v_1, v_2) \in E$ and its corresponding type l_e **do**
3: Predicate name $p \leftarrow l_{(v_1,v_2)}$
4: Fact $f \leftarrow p(v_1, v_2)$
5: $D \leftarrow D \cup \{f\}$
6: **end for**
7: Set facts of the target predicate π in D as positive examples
8: **return** D