ONTOLOGY MAPPING NEURAL NETWORK: AN APPROACH TO LEARNING AND INFERRING CORRESPONDENCES AMONG ONTOLOGIES

PhD Dissertation

by

Yefei Peng

B.S., Tsinghua University, 1999

M.S., Purdue University, 2004

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Yefei Peng

It was defended on

April 9th 2010

and approved by

Dr. Paul Munro, School of Information Sciences

Dr. Daqing He, School of Information Sciences

Dr. Hassan Karimi, School of Information Sciences

Dr. Bambang Parmanto, Department of Health Information Management

Dr. Michael Spring, School of Information Sciences

Dissertation Director: Dr. Paul Munro, School of Information Sciences

ONTOLOGY MAPPING NEURAL NETWORK: AN APPROACH TO LEARNING AND INFERRING CORRESPONDENCES AMONG ONTOLOGIES

Yefei Peng, PhD

University of Pittsburgh, 2010

An ontology mapping neural network (OMNN) is proposed in order to learn and infer correspondences among ontologies. It extends the Identical Elements Neural Network (IENN)'s ability to represent and map complex relationships. The learning dynamics of simultaneous (interlaced) training of similar tasks interact at the shared connections of the networks. The output of one network in response to a stimulus to another network can be interpreted as an analogical mapping. In a similar fashion, the networks can be explicitly trained to map specific items in one domain to specific items in another domain. Representation layer helps the network learn relationship mapping with direct training method.

The OMNN approach is tested on family tree test cases. Node mapping, relationship mapping, unequal structure mapping, and scalability test are performed. Results show that OMNN is able to learn and infer correspondences in tree-like structures. Furthermore, OMNN is applied to several OAEI benchmark test cases to test its performance on ontology mapping. Results show that OMNN approach is competitive to the top performing systems

that participated in OAEI 2009.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I'm heartily thankful to my advisor, Dr. Paul Munro, for his supervision, guidance and support through out my Ph.D. study, and for allowing me to work in my own way.

I gratefully acknowledge the members of my committee, Dr. Daqing He, Dr. Hassan Karimi, Dr. Paul Munro, Dr. Bambang Parmanto and Dr. Michael Spring for their effort and time in guiding me to fulfill the requirement of the degree.

I am deeply grateful to my parents, sisters, and their families. Their expectation encouraged me to persevere till the end. I owe my loving thanks to my wife Ming Mao, for her dedication, love and persistent confidence in me all the time, also for her insightful discussions on my research. I would like to thank my daughters, Jing and Ann, for allowing me to sacrifice play time with them.

The school of Information Sciences has been a warm place provided the support I needed to complete my study. All the faculties and staffs have been very friendly and helpful.

The financial support from the School of Information Sciences and Yahoo! is gratefully acknowledged.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

1.0 INTRODUCTION

1.1 INTRODUCTION

The vision of the Semantic Web [BLHL01] provides many new perspectives and technologies to overcome the limitation of the WWW. Ontologies are a key component to solving the problem of semantic heterogeneity, and thus enable semantic interoperability between different web applications and services. An ontology is a formal, explicit specification of a shared conceptualization [Gru93b] that provides precise notations and explicit meanings of data (i.e. semantics) in a domain [UG04]. Although it is ideal to have a single or even a small set of shared ontologies that can be accepted by all or be used as a reference to derive people's own ontologies, such a utopia is unrealistic.

The task of finding semantic correspondences between similar elements of different ontologies is known as ontology mapping or ontology matching. In this area, different authors "use different words to refer to similar concepts, and, vice versa, sometimes different concepts are referred to by the same name" [ES07]. For example, ontology mapping is defined as the process of "mapping concepts in the various ontologies to each other, so that a concept in one ontology corresponds to a query (i.e. view) over the other ontologies" [CGL01]. Ontology matching is defined as "finding semantic mappings between ontologies" [DMDH02].

In the rest of the dissertation, the term of ontology mapping refers to the task of finding semantic correspondences between similar elements of different ontologies. The following applications illustrate how semantic correspondences are required in different scenarios, as well as emphasize the importance of ontology mapping.

First of all, ontology mapping is important to the emerging Semantic Web. The pervasive usage of agents and web services is a characteristic of the Semantic Web. However agents might use different protocols that are independently designed, which means that when agents meet they have little chance for them to understand each other without an "interpreter". Therefore ontology mapping is "a necessary precondition to establish interoperability between agents or services using different ontologies" [Ehr06]. It provides the possibility for agents/services to either translate their messages or integrate bridge axioms into their own models [vEdBvdHM01].

Ontology mapping is also widely used to support information transformation and integration [DMQ05] [CM03] [NM03].

Moreover, ontology mapping can support query processing across disparate sources by expanding or rewriting the query using the correspondent information between multiple ontologies [GKD97] [CGL01] [HIST03] [MKSI96] [GH05]. The application of ontology mapping can also be found in generating an ontology extension [DMQ03] and many other scenarios [EBB$^+$04].

The importance of ontology mapping in different applications motivates our research interest in this area. The ultimate goal of this project is to accelerate the success of the semantic interoperability between different information systems in the WWW. More specifically, we aim to explore a new generic method to automatically map ontologies with minimum human effort because while ontologies can be mapped either by hand or by tools, manual mapping becomes impractical as the complexity and volume of ontologies increase. Alternatively developing fully or semi-automated mapping algorithms/tools has attracted the interest of researchers in various areas [Noy04] [NDH05] [RB01].

1.2 OVERVIEW OF OMNN APPROACH

The Ontology Mapping Neural Network (OMNN) extends the Identical Elements Neural Network(IENN)'s [Mun96, MB05, BM06, Mun08] ability to represent and map complex relationships. The network can learn high-level features common to different tasks, and use them to infer correspondence between the tasks. The learning dynamics of simultaneous (interlaced) training of similar tasks interact at the shared connections of the networks. The output of one network in response to a stimulus from another network can be interpreted as an analogical mapping. In a similar fashion, the networks can be explicitly trained to map specific items in one domain to specific items in another domain.

Separate relationship input layers and the relationship representation layer that connect to both input layers can map relationships among different ontologies. The representation layer helps the network learn relationship mapping with an explicit training method.

Figure 1.1 shows system architecture of OMNN ontology mapping approach. When used as ontology mapping approach, the OMNN network has to be trained with preliminary mappings. The preliminary mapping part utilizes textual information to generate preliminary mappings. High confident mapping pairs will be used as cross-training data for OMNN. After training, OMNN will be cross-tested to generate a node response matrix and a relationship distance matrix. Multiple simulations will be run to get average matrixes, which will improve performance in case of simulations converge to a local minimum with bad result. A simple mapping extraction algorithm is run over the matrixes to extract node mappings and relationship mappings.

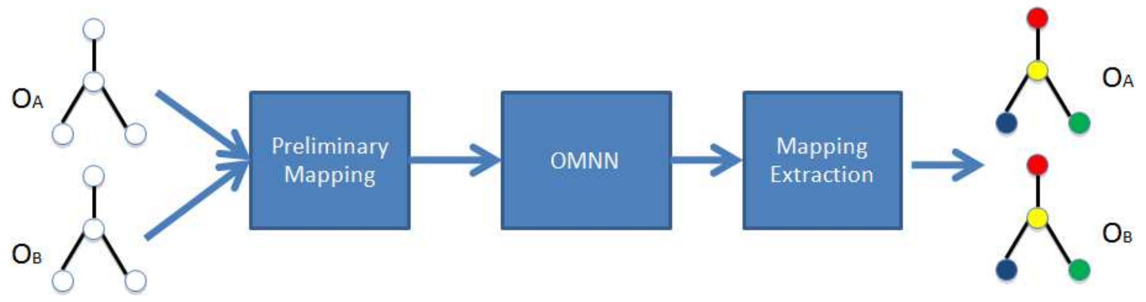


Figure 1.1: System architecture of OMNN

1.3 CONTRIBUTIONS OF THE DISSERTATION

The contributions of the dissertation can be summarized as follows:

1. Proposal of a new neural network architecture OMNN for graph mapping, it supports not only item mapping, but also relationship representation and mapping.
2. Proposal of a novel explicit training method for relationship mapping.
3. Deployment of a new ontology mapping approach with a novel way of combining textual information and structural information.
4. Evaluation of the OMNN approach with comparison with interesting approaches, OMNN is expected to be competitive with existing approaches, and the expectation is confirmed by the evaluation.

1.4 OUTLINE

The rest of this dissertation is organized in the following manner. Chapter 2 reviews representative works by different researchers for ontology mapping. Chapter 3 describes OMNN approach for graph mapping and integrated OMNN approach for ontology mapping. Chapter 4 evaluates OMNN graph mapping approach on family tree test cases. Results show the OMNN approach performs as expected. Chapter 5 gives out the experimental results of OMNN ontology mapping approach on the benchmark tests from OAEI campaign 2009, and compares it with other systems from OAEI 2009. Results show OMNN approach is competitive to other systems. Chapter 6 summarizes our work and outlooks future work.

2.0 LITERATURE REVIEW

In the first part of this chapter, main methods of ontology mapping from literatures are reviewed. Graph-based methods, machine learning methods, heuristic and Rule-based methods are the three main categories of mapping methods. Latest ontology mapping systems from OAEI 2009 are also reviewed. These systems represent most recent state of the art ontology mapping systems. Since our approach is neural network based, in the last part of this chapter, related neural network models are reviewed.

2.1 STATE OF THE ART ONTOLOGY MAPPING APPROACHES

2.1.1 Graph-based Methods

2.1.1.1 Anchor-PROMPT Anchor-PROMPT [NM01], which extends PROMPT [NM00], is an ontology merging and mapping tool with a sophisticated prompt mechanism for term matching. It treats an ontology as a directed labeled graph, where concepts are nodes and relations are arcs.

The Anchor-PROMPT algorithm is based on the observation that if two pairs of terms are similar and paths connect each pair, then the elements in these paths are often similar as well.

A set of anchors that are identified manually by the user or automatically generated are then served as input to the system. Anchor-PROMPT traverses the paths between the anchors and computes the terms along these paths to find similar terms. Finally, the Anchor-PROMPT produces a set of pairs of semantically related terms.

Figure 2.1 shows one example. There are two pairs of pre-identified anchors, classes A and B and classes H and G, and two parallel paths, one from A to H in Ontology 1 and the other from B to G in Ontology 2. The Anchor-PROMPT traverses the two paths and increments the similarity score between each two classes (i.e., classes C and D, classes E and F) reached in the same step. Then Anchor-PROMPT repeats the process for all of the existing paths that start and end at the anchors and cumulatively aggregates the similarity scores.

The evaluation shows that 75% of the precision of ontologies is developed independently by different groups of researchers. One obvious limitation with anchor-PROMPT is that it does not work well when the hierarchies of two ontologies are different, i.e. one has more layers than the other. The other limitation is that Anchor-PROMPT is time-consuming. Its worst case run-time behavior is $O(n^2log(n))$, compared to PROMPT $O(nlog(n))$, GLUE of $O(n^2)$ and QOM $O(nlog(n))$.

2.1.1.2 Similarity Flooding Similarity Flooding [MGMR02] is a generic graph matching algorithm based on a fixpoint computation. The algorithm takes two graphs (schemas, catalogs, or other data structures) as input, and produces a mapping between corresponding nodes of the graphs as output. The principle of the similarity flooding algorithm is that the similarity between two nodes depends on the similarity between their adjacent nodes. The spread of similarities is similar to how IP packets flood a network in broadcast communication. This is why the algorithm is called similarity flooding.

Figure 2.2 illustrates the Similarity Flooding algorithm. Two ontologies, A and B, are represented by directed labeled graphs based on the Open Information Model specification. The algorithm creates another graph whose nodes, i.e., (a, b) and (a_1, b_1), are pairs of nodes of the initial two graphs, and there is an edge l_1 between (a, b) and (a_1, b_1) whenever there are edges (a, l_1, a_1) in the first graph and (b, l_1, b_1) in the second one. Initial similarity values between nodes are calculated based on their labels. Similarities are then re-computed between nodes as a function of the similarity between the adjacent nodes. The process is iterated until converge.

After the algorithm runs, a human is expected to check and adjust the results. The algorithm is evaluated by counting the number of adjustments made by the human.

Thus, the function of the accuracy used in their evaluation is:

$$Accuracy = Recall \times (2 - \frac{1}{Precision})$$

This definition shows that the notion of the accuracy only makes sense when precision is not less than 0.5. If more than half of the mappings are wrong, it would take the user more effort to remove the false positives and add the missing mappings than to do the mapping manually from a scratch. Their evaluation shows that their mapping accuracy over seven users and nine problems averaged 52%.

Similarity Flooding algorithm is a generic graph matching algorithm. Though the Similarity Flooding algorithm can be applied to 1-to-n mapping, it obtains this feature by decreasing the threshold of similarity. That is not a real 1-to-n mapping.

Some limitations of the Similarity Flooding algorithm are:

1. The algorithm is based on the assumption that adjacency contributes to similarity propagation. Thus, when adjacency information is not preserved the algorithm will perform unexpectedly. For example, in some HTML pages, sometimes nodes that are displayed visually close to each other are structurally far apart from each other. In another case two cells in an HTML table vertically adjacent could be far apart in the document and not able to contribute to similarity propagation.
2. The algorithm can only be applied to equal-typed models. The meaning of the edges in the two models must be similar. For instance, it works when mapping an XML schema against another XML schema, but not when mapping a relational schema against an XML schema.

2.1.2 Machine Learning Methods

GLUE [DMDH03] is a system that employs machine learning techniques to find ontology mappings. If given two ontologies, for each concept in one ontology, GLUE finds the most similar concept in the other ontology. GLUE works with several similarity measures that are defined with probabilistic definitions. Multiple learning strategies exploit different types of information from instances or taxonomy structures. GLUE can also use common sense knowledge and domain constraints instead of relaxation labeling. It is a well-known constraint optimization technique adapted to work efficiently.

Figure 2.3 shows the architecture of GLUE. The Distribution Estimator, Similarity Estimator, and Relaxation Labeler module are the three main modules. The Distribution

Estimator takes two taxonomies and their data instances as input. Machine learning techniques are then applied to compute joint probability distribution for every pair of concepts. Not a single learner but a set of base learner and meta learner are used in this step. The Similarity Estimator combines the joint probability distribution from Distribution Estimator with a user-supplied similarity function; the result is a similarity matrix with a similarity value for each pair of concepts. The Relaxation Labeler module then takes the similarity matrix, domain-specific constraints and heuristic knowledge, and searches for the mapping configuration that best satisfies the domain constraints and the common knowledge by taking into account the observed similarities. This mapping configuration is the final output of GLUE.

The problem of GLUE is that it requires a large number of instances associated with the nodes in taxonomies. These are not available in most ontology mapping situations. Actually, it's not only problem of GLUE; all machine learning based systems have this limitation.

2.1.3 Heuristic and Rule-based Methods

2.1.3.1 PROMPT PROMPT[NM00] is an algorithm that provides a semi-automatic approach to ontology merging and alignment. Figure 2.4 shows the workflow of PROMPT algorithm: the gray boxes indicate the actions performed by PROMPT and the white box indicates the action performed by the user. To make the initial suggestions, PROMPT uses a measure of linguistic similarity among items in ontologies. PROMPT automatically creates some preliminary mapping suggestions by using a mix of linguistic similarity and structure similarity. The user is then guided to make judgments on each of the suggestions. Whenever user makes some judgment, PROMPT will automatically update the suggestions, determines possible inconsistencies in the state of the ontology based on the userŠs actions, and suggest ways to remedy these inconsistencies. For instance, if a user says that two classes in two source ontologies are the same, that means these two ontologies should be merged. PROMPT would then analyz the properties of these classes, their subclasses and superclasses to look for similarities of their definitions and suggest additional correspondences. The strategies and algorithms are based on a general OKBC-compliant knowledge model. Therefore, these results are applicable to a wide range of knowledge representationand ontology-development systems.

The formative evaluation shows that PROMPT is very effective. A human expert followed 90% of the suggestions that PROMPT generated and that 74% of the total knowledge-base operations invoked by the user were suggested by PROMPT. One limitation of PROMPT is that it determines the similarity based on the exact equality of labels and only one similarity is computed. No similarity aggregation is performed.

2.1.3.2 QOM QOM (Quick Ontology Mapping)[ES04] is based on the hypothesis that mapping algorithms can be streamlined so that the loss of quality is marginal, but the improvement of efficiency is tremendous for the ad-hoc mapping of large size, light weight ontologies. It is defined by the process model shown in Figure 2.5, it is started with two ontologies, which are to be mapped onto one another, as its input.

1. Feature Engineering transforms the initial representation of ontologies into a format digestible for the similarity calculations. For instance, the subsequent mapping process may only work on a subset of RDFS primitives. This step may also involve complex transformations, e.g. it may require the learning of classifiers as input to the next steps.

2. Search Step Selection. The run time complexity of a mapping algorithm is directly related to the number of candidate mapping pairs to be examined. In the Search Step Selection process, QOM applies a heuristic method that makes use of ontological structures to reduce the number of candidate mappings, which improves the efficiency of QOM.
3. In the Similarity Computation step, the similarity between entities of ontologies is generated by using various similarity functions and heuristics. QOM avoids the complete pair-wise evaluation of ontology trees to improve efficiency.
4. Similarity Aggregation. In this step, similarity measures from different methods are aggregated. QOM uses sigmoid function to transform similarities first, and then calculate average of similarities. The sigmoid function will emphasizes high individual similarities rather than low individual similarities.
5. Interpretation. Three methods exist to interpret similarity results. In the first method, spurious evidence of similarity is filtered out by a threshold. The second method is relaxation labeling. The third method combines structural and similarity criteria.
6. Iteration. Through several iterations the quality of the results rises considerably. QOM limits the number of iterations to 10 because empirical findings indicate that further iterations produce insignificant changes. Eventually, the output is a mapping table representing the relationship between the two ontologies.

Depending on the scenario, QOM can be very effective and efficient, reaching high quality levels quickly by a factor of 10 to 100 times compared to approaches such as PROMPT and NOM (a simulation of Anchor-PROMPT algorithm). One problem of QOM is that its optimization of mapping approach decreases the overall mapping quality. Therefore, QOM is only recommended when ontologies are large-scaled.

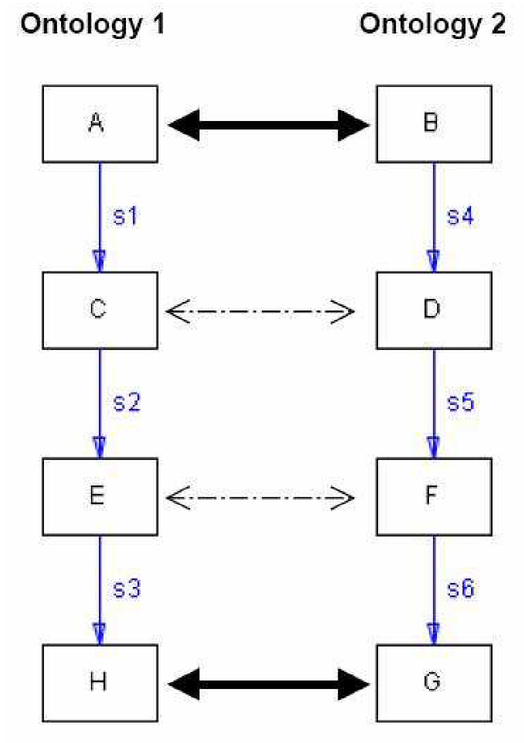


Figure 2.1: Anchor-PROMPT. Traversing the paths between anchors.

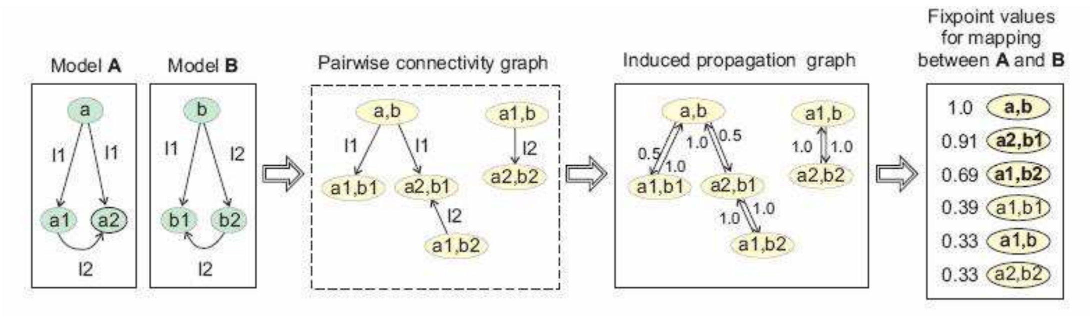


Figure 2.2: Example of similarity flooding

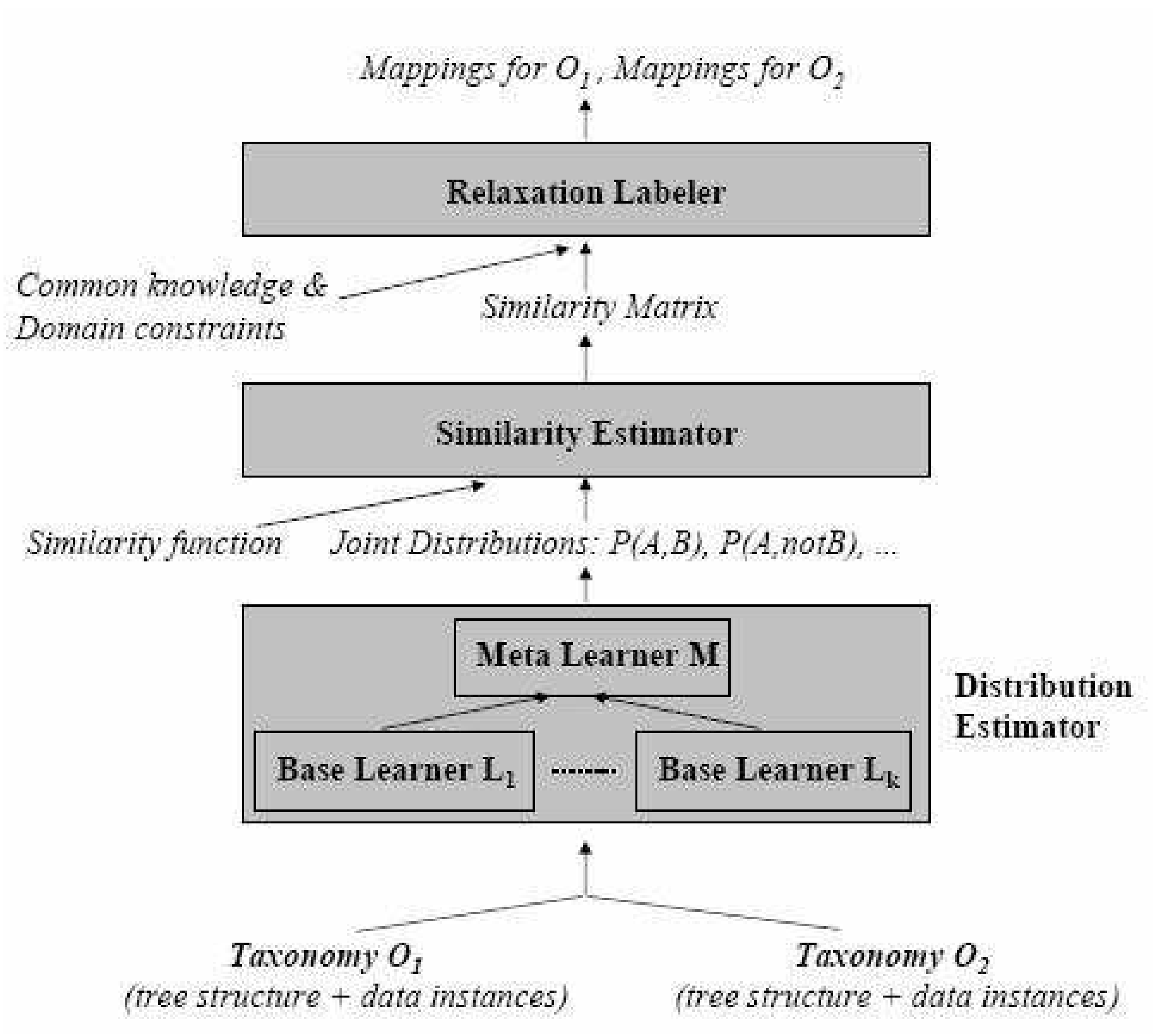

Mappings for O_1, Mappings for O_2
Relaxation Labeler
Common knowledge & Domain constraints
Similarity Matrix
Similarity Estimator
Similarity function
Joint Distributions: P(A,B), P(A,notB), ...
Meta Learner M
Base Learner L_1
Base Learner L_k
Distribution Estimator
Taxonomy O_1 (tree structure + data instances)
Taxonomy O_2 (tree structure + data instances)

Figure 2.3: Architecture of GLUE

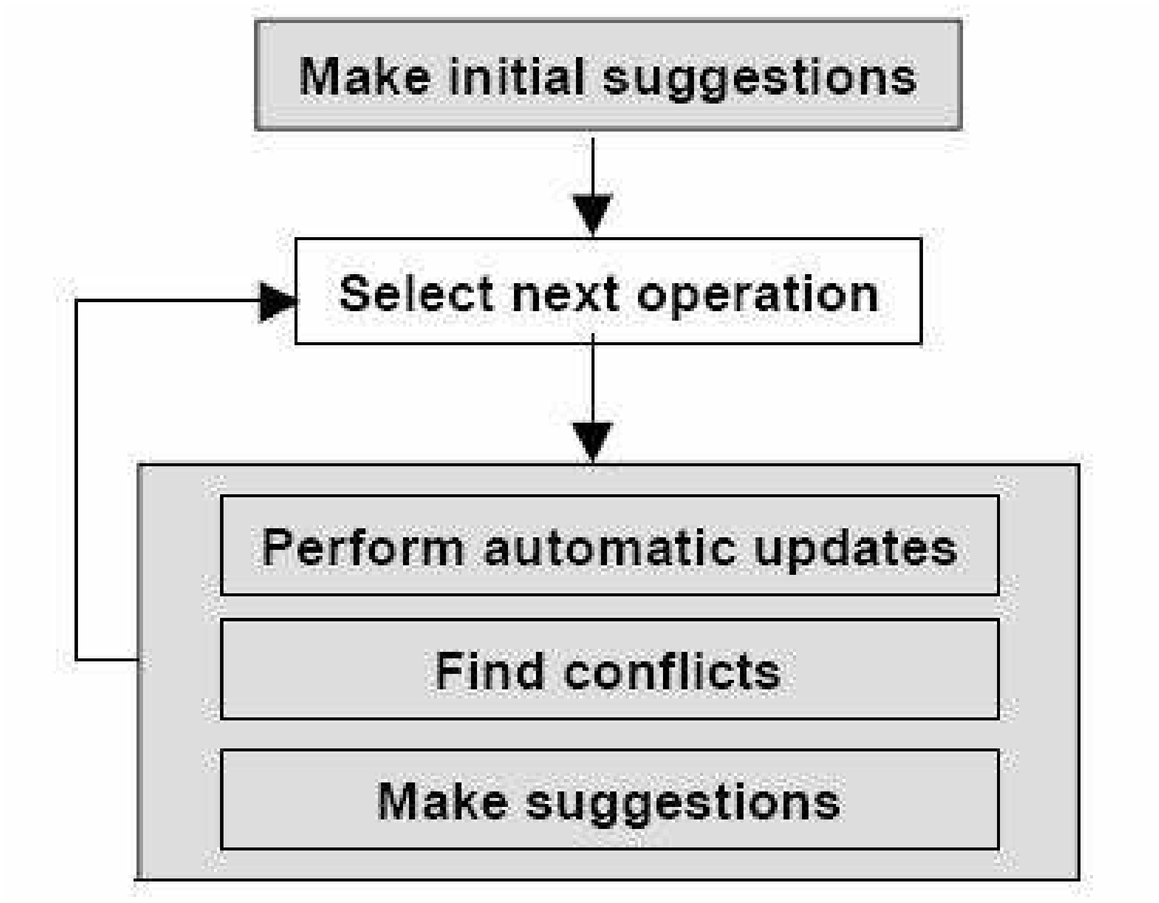


Figure 2.4: Workflow of PROMPT algorithm

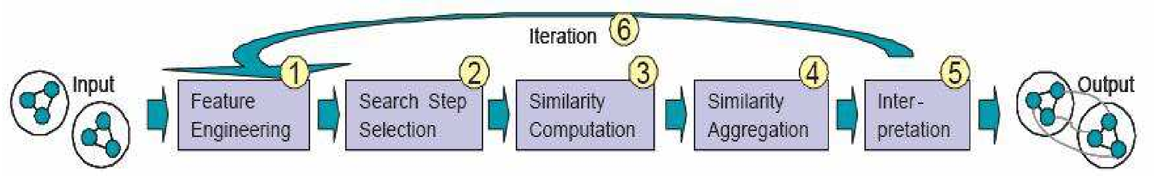


Figure 2.5: QOM mapping process

2.2 THE LATEST ONTOLOGY MAPPING SYSTEMS

Formed in 2004, The Ontology Alignment Evaluation Initiative (OAEI)[1] is a coordinated international initiative to forge evaluations of ontology mapping methods. OAEI organizes a yearly evaluation event and publishes the tests and results of the event for further analysis. Starting from 2006, OAEI campaign is associated with ISWC Ontology matching (OM) workshop each year. The most recent OAEI campaign is OAEI 2009. RiMOM[ZZS+09], LILY[WX09], ASMOV[JMSK09] and aFlood[HA09] are four top ranked systems from this campaign.

2.2.1 RiMOM

RiMOM[ZZS+09] is a general ontology mapping system based on Bayesian decision theory. It utilizes normalization and NLP techniques and integrates multiple strategies for ontology mapping. Afterwards RiMOM uses risk minimization to search for optimal mappings from the results of multiple strategies. The process of RiMOM is shown in Figure 2.6.

There are six major steps in a general alignment process of RiMOM.

1. Ontology Preprocessing and Feature Factors Estimation.
2. Strategy Selection. Multiple strategies utilize different features extracted from input ontologies. The basic idea of strategy selection is that if some features extracted from two ontologies are very similar, then strategies based on these features will be given high weight. A feature factor is calculated for each feature; the weight of the strategy is decided by the feature factors. If a feature factor is too low, the strategy using the

[1]http://oaei.ontologymatching.org/

feature may not be used at all.

3. Single strategy execution. The selected strategies are run to find the alignment. Each strategy outputs an alignment result.
4. Alignment combination. In this phase RiMOM combines the alignment results obtained by the selected strategies. The combination is conducted by a linear interpolation method.
5. Similarity propagation(Optional). In this step, structure information is used to refine alignments and find new alignments.
6. Alignment refinement. Several heuristic rules are used to further refine the alignments from the previous steps. The main purpose is to remove unreliable alignments.

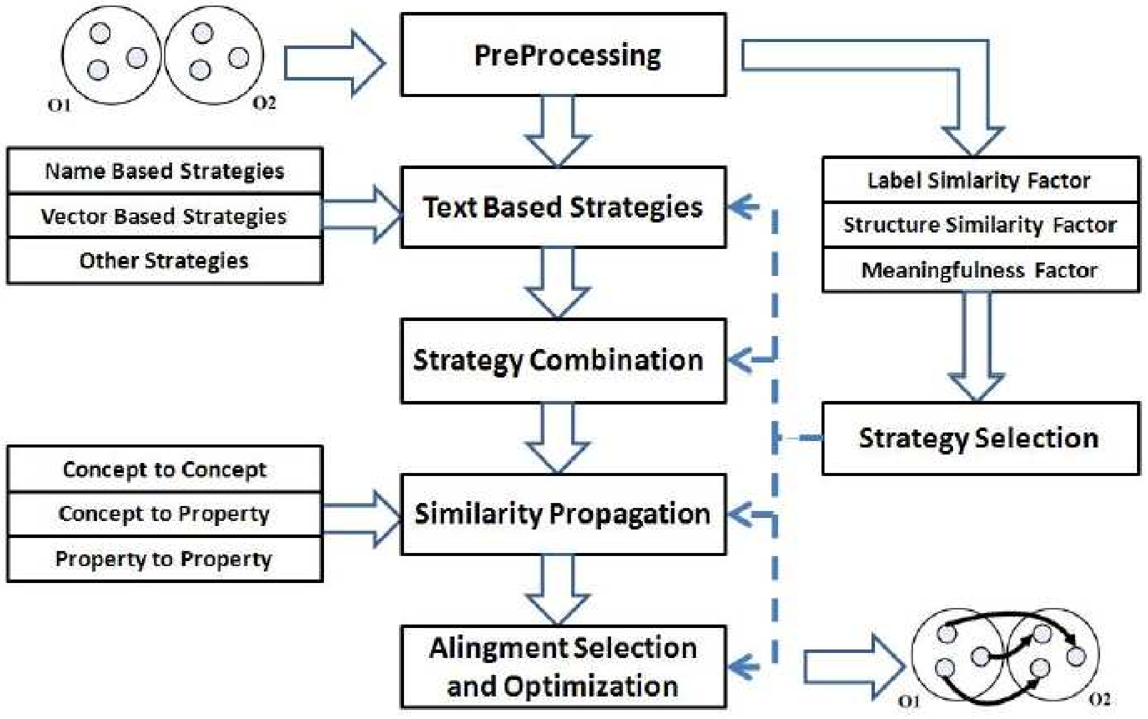


Figure 2.6: System architecture of RiMOM

2.2.2 LILY

LILY[WX09] is a generic ontology mapping system based on the extraction of semantic subgraph. It exploits both linguistic and structural information in semantic subgraphs to generate initial alignments. After that, a subsequent similarity propagation strategy is applied to produce more alignments if necessary. Finally, LILY uses the classic image threshold selection algorithm to automatically select the threshold, and then extracts final results based on the stable marriage strategy.

LILY has different functions for different kinds of tasks: for example, Generic Ontology Matching method (GOM) is used for common matching tasks with small size ontologies; Large scale Ontology Matching method(LOM) is used for matching tasks with large size ontologies; Semantic Ontology Matching method (SOM) is used for discovering the semantic relations between ontologies.

Two limitations of LILY are that it needs the user to manually set the size of subgraph according to different mapping tasks and the efficiency of semantic subgraph is very low in large-scale ontologies.

2.2.3 ASMOV

Figure 2.8 illustrates the fully automated ASMOV [JMSK09] mapping process. In the preprocessing phase, the ontologies are loaded into memory using the Jena ARP parser and ASMOV's ontology modeling component. Lexical similarity measures are calculated for each pair of concepts, properties and individuals. Specifically, four features are analyzed: textual description (id, label, and comment), external structure (parents and children), internal structure (property restrictions for classes; types, domains, and ranges for properties),

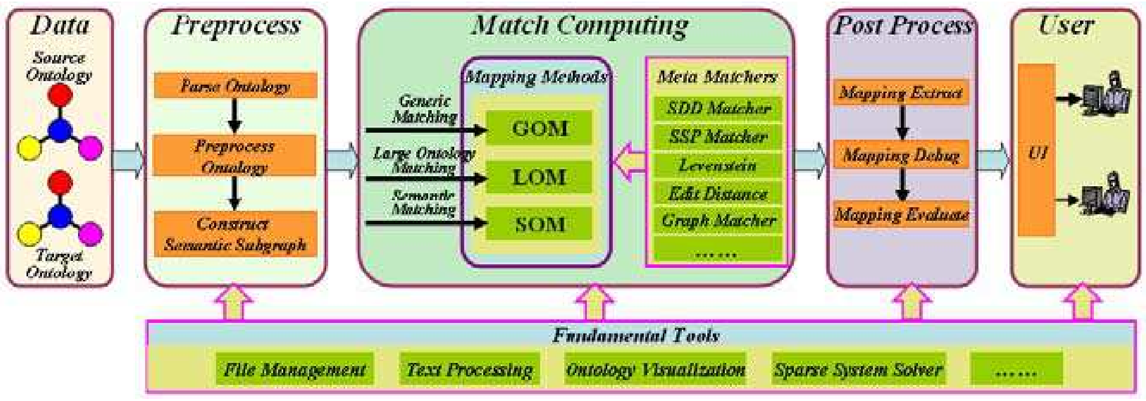


Figure 2.7: System architecture of LILY

and individual similarity. The similarities between pairs of entities along the relational structure, internal structure, and extensional dimensions are calculated, and an overall similarity measure (or confidence value) is stored. Then a step of semantic verification is performed to detect and prune some inconsistent mappings. Five types of inconsistencies are detected: multiple entity correspondences, Crisscross correspondences, Disjointness-subsumption contradiction, disjointness-subsumption contradiction, and domain and range incompleteness.

2.2.4 AFlood

AFlood[HA09] is an ontology schema matching algorithm that takes the essence of the locality of reference by considering neighboring concepts and relations to align the entities of ontologies. The algorithm of aFlood is shown in Figure 2.9. It starts off a seed point called an anchor (a pair of Şlook-alikeŤ concepts across ontologies). Then small blocks of concepts and related properties are dynamically collected considering neighborhood information from the anchor points. Local alignment process will align the small blocks using lexical and

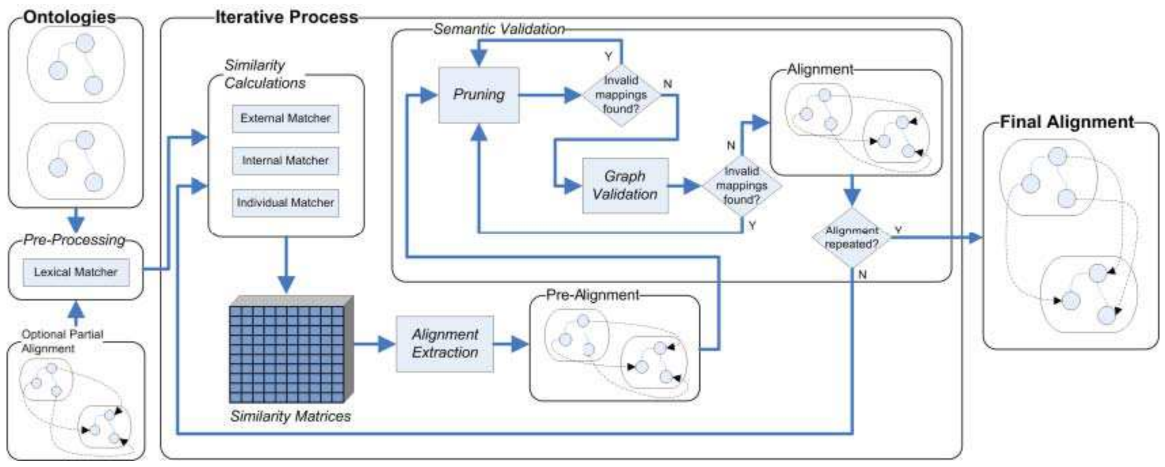


Figure 2.8: System architecture of ASMOV

strutual information. Aligned pairs are considered the new anchors. The process is then repeated until all of the anchors are processed.

The strength of aFlood is that it reduces the number of comparisons between entities by only matching small blocks not whole ontologies, which increases efficency.

2.3 RELATED NEURAL NETWORK MODELS

The OMNN approach described in this dissertation is inspired by works about mapping structure analogy. In the classic paper, Hinton [Hin86] demonstrates the role of internal representations in the network solution of a "family tree" task. In that paper, the network is trained on triples of the form <agent, relation, patient> to generate the patient, given the agent and relation. For example, given the input agent="Colin", and relation="mother", the network should compute an output that is a representation of Victoria (Colin's Mother).

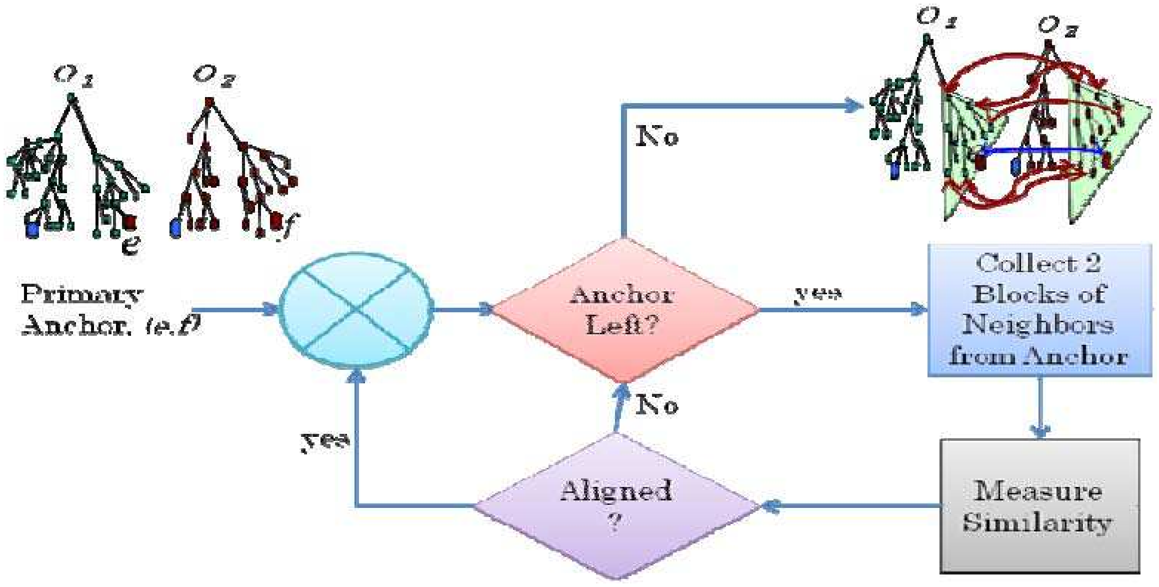


Figure 2.9: System architecture of aFlood

The network is trained to learn family relations from two disjoint family trees with identical structures, i.e., there is a one-to-one mapping between individuals in two domains. Since the output units are trained to distinguish between the two, the hidden unit weights to the input layer are identical for many homologous pairs, and are exactly opposite for others. While Hinton's paper does not address analogy, similarities in the hidden unit representations of the homology are apparent.

The Identical Elements Neural Network (IENN)[Mun96, MB05, BM06, Mun08] approach developed by Munro is that a network that can learn high-level features common to different tasks that are encoded in weight between two hidden layers. The shared weights then could be used to learn a novel task which has similar high-level attributes with the trained tasks. This IENN approach uses simultaneous training of multiple tasks, but differs from MTL in that the tasks do not share a common input. In principle, the tasks can have inputs that have very different coding, dimensionality, etc. Experiment results show that the shared weights significantly accelerate the learning of the third task more than using random initial weights. The network can also find analogical correspondence between patterns of tasks. The IENN network faces difficulty when representing complex structures and relationships. The only relationship embedded is “neighborhood”; “neighbors” and “self” are represented as target at output layers.

The IENN network was later extended by the author to include a shared input subvector, and was applied to tree mapping problem [PM09, MP09]. The network extends the ability of the IENN by adding an input relationship subvector so that it can represent complex relationship between input patterns. The network supports theoretically any number of relationships. Four relationships were used to represent tree structure: “self”, “parent”, “child”, and “sibling”.

The network architecture proposed in this dissertation can be seen as an offspring of the IENN[Mun96, MB05, BM06, Mun08] and its extension [PM09, MP09]. The purpose of this research is to deeply understand how the network works, how it can be applied to graph mapping, and whether it works for ontology mapping.

3.0 OMNN APPROACH

In 3.1, two new network architectures are described for graph mapping problems. The first network architecture provides a way to separate representation of concepts and relationships. Thus, it'll be easier to achieve a deeper understanding of how the network works by analyzing the representation of concepts and relationships that evolve during training. On top of the first network architecture, the second network architecture splits the relationship input layer into two layers, so as to represent and map different relationships from two graphs.

In 3.2, the two network architectures are integrated into an ontology mapping approach. The integrated approach combines both linguistic information and structural information for mapping ontologies.

In 3.3, test cases and evaluation criteria are described. Family tree test cases are outlined for graph mapping approach, and OAEI benchmark test cases are described for an ontology mapping approach.

3.1 GRAPH MAPPING

3.1.1 Introduction

A key element in abstract reasoning is the ability to recognize corresponding elements, features, and relationships in different situations that may not appear similar on the surface. Here, the term structural mapping is used to refer to this aspect of cognition. Let a *semantic graph* S be defined as a set of items I and a set of relationships R among them.

$$S = \{I^S, R^S\}$$

$$I_S = \{i_1^S, i_2^S, ..., i_{N_s}^S\}$$

$$R_S = \{r_1^S, r_2^S, ..., r_{M_s}^S\}$$

$$r_k^S = r_k^S(i_1^S, i_2^S, ...)$$

And let a *structural mapping* M from one system to another be a specification of corresponding items and relationships.

$$M(A, B) = ((i_{j1}^A, i_{j1}^B), (i_{j2}^A, i_{j2}^B), ...; (r_{k_1}^A, r_{k_1}^B), (r_{k_2}^A, r_{k_2}^B)...)$$

$$or\ i_j^B = m(i_j^A)\ and\ r_k^B = m(r_k^A)$$

A structural mapping is considered a structural analogy to the extent that corresponding items are related by corresponding relationships as in following equation:

$$r_k^B(i_{k_1}^B, i_{k_2}^B, ...) = m(r_k^A(m(i_{k_1}^A), m(i_{k_2}^A), ...))$$

In this dissertation, only structural information is used for graph mapping. However, OMNN ontology mapping (3.2) approach can be used when textual information is available as well.

The IENN[Mun96, MB05, BM06, Mun08] shown in Figure 3.1 has at least one limitation. Only one type of relationship can be represented. It associates input of one subnet to target at output of corresponding subnet in vertical task. In the use of IENN, self node and neighborhood nodes are served as a target. Thus, the relationship represented is a mixture of self and neighborhood. This restricts the representation power of IENN; it cannot represent any more relationships.

Here two network architectures are proposed.

3.1.2 Network Architecture 1: Adding Relationship and Representation Layers

3.1.2.1 Network Structure The first network architecture is applied to cases when two graphs have the same set of relationships, and one to one relationship mappings are known. Real world examples include taxonomy mapping when the only relationship is "is-a", tree graph mapping with "parent" and "child" relationships, as well as ontology mappings where the relationships are same between two ontologies and their mappings are known.

The first proposed network architecture is shown in Figure 3.2. A_{in} and B_{in} are input subvectors for nodes from graph A and graph B respectively. S_{in} is the shared subvector representing the relationship in both graphs. It's similar to IENN. With IENN, the difference is that a shared subvector S_{in} can represent multiple relationships.

In this network, each to-be-mapped node in graph is represented by a single active unit in input layers (A_{in}, B_{in}) and output layers (A_{out}, B_{out}), the same way as in IENN. For relationships representation in input layer (S_{in}), each relationship is represented by a single active unit.

Two major differences between OMNN architecture are shown in Figure 3.2 and IENN.

First, relationship input layer is added, so that multiple relationships are supported. Second, representation layer for items (AB_r) and relationships (S_r) are added. In IENN, the representation of items and relationships has to be decided. The representation is encoded into patterns of input layer. In IENN, each item is represented by a single active unit in input layer. One can argue that the first hidden layer has a representation of input items, but it's actually a mixed representation of items and relationships. Pure representation of items or relationship nodes cannot be found. In contract, representation layers are added in OMNN so that items and relationships have their own representation layers.

Although Figure 3.2 only shows two input subvectors and two output subvectors, it's obvious that it can be generalized to multiple input subvectors and multiple output subvectors, so that mapping among multiple graphs can be done.

3.1.2.2 Training

The network shown in Figure 3.2 has multiple sub networks. To name a few, we'll use:

- Net_A : $\{A_{in}\text{-}AB_r\text{-}X_{AB};\ S_{in}\text{-}S_R\text{-}X_S\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_A\text{-}A_{out}$;
- Net_B : $\{B_{in}\text{-}AB_r\text{-}X_{AB};\ S_{in}\text{-}S_R\text{-}X_S\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_B\text{-}B_{out}$;
- Net_{AB} : $\{A_{in}\text{-}AB_r\text{-}X_{AB};\ S_{in}\text{-}S_R\text{-}X_S\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_B\text{-}B_{out}$;
- Net_{BA} : $\{B_{in}\text{-}AB_r\text{-}X_{AB};\ S_{in}\text{-}S_R\text{-}X_S\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_A\text{-}A_{out}$;

There are two types of training in the network: vertical training and cross training. Next we'll use the graphs shown in Figure 3.4 as examples to describe the two types of training.

Vertical training means input and output banks belong to same tasks, e.g. standard back-propagation training on Net_A or Net_B. Vertical training is used to train the network to learn each single graph. The training data is from one graph only, for example, "parent" (Relation) of “Bus” (Subject) is “Vehicle” (Object). In this example, since it’s part of graph A, Net_A will be used. The representation of “Bus” will be activated at A_{in}; the representation of “parent” (Relation) will be activated at S_{in}, and the target at A_{out} will be the representation of “Vehicle” (Object).

Cross training means input and output banks belong to different tasks, e.g. standard backprop training on Net_{AB} or Net_{BA}. Cross training is used to enforce correspondence between a specific item in graph A and a specific item in graph B; for example, “self” of “Vehicle” in graph A corresponds to “Automobile” in graph B. Net_{AB} will be used to train this record. The representation of “Vehicle” will be activated at A_{in}; the representation of “self” (Relation) will be activated at S_{in}; the target at B_{out} will be the representation of “Automobile” (Object).

Networks are initialized by setting the weights to small random values from a uniform distribution. The network is trained with two vertical training tasks (Net_A and Net_B), and two cross training tasks (Net_{AB} and Net_{BA}). One training cycle of the networks is:

1. Randomly train a record for Net_A
2. Randomly train a record for Net_B
3. Train a record for Net_{AB} and the corresponding record for Net_{BA} with a probability.

3.1.2.3 Testing During training and testing, patterns are only presented on one input bank(A_{in} or B_{in}) and shared bank(S_{in}). This restriction maintains a consistent net input to the units of H_1. There is no reason for such a constraint among the output layers during testing. Thus, output patterns across all banks can be examined in response to the input on a single input bank. This makes a correspondence analysis between items in different tasks possible. That is, the output in A_{out} generated by an A_{in} can be compared to the outputs in B_{out} generated by the set of B_{in} inputs. Any B_{in} input is a candidate for the image of A in the input space of B_{in} when the B_{in} input generates a B_{out} output that is sufficiently close to the one generated by A_{in}.

Another approach to correspondence analysis is to compare representations of inputs from different input banks at the AB_r level. If an A_{in} pattern and a B_{in} pattern have identical or nearly identical) AB_r representations, then they must give the same (or nearly the same) output patterns on all output banks.

In this study, the cross testing method is presenting patterns on one input bank and shared bank and examining output bank of another graph. The relationship presented at shared bank is “self” only. For example, to map node in graph A to graph B, input pattern of this node in graph A is presented on A_{in}, “self” relationship is presented on S_{in}, then output at B_{out} contains information about the mapping. If we want to map node in graph B to graph A, input pattern of this node in graph B is presented on B_{in}; “self” relationship is presented on S_{in}, and then output at A_{out} contains information about the mapping. In our experiment, we only use “self” relationship in cross-testing. It’s possible that using other relationship will produce different mapping configurations. We’ll leave that for future work.

Since neural network training is subject to local minimum, running only one simulation may not produce a good performance, and it may not reflect the true performance of the approach. Running multiple simulations and getting the average of them is a better method to fairly evaluate the approach. For each node to test in one graph, say graph A, cross-testing is performed and output for the other graph (B_{out}) is recorded. Output for all nodes in graph A can be combined into a matrix. In this matrix, each row represents one node in graph A and each column represents one node in graph B. For each simulation, we have such a matrix. We obtain a final mapping matrix by averaging all matrices from all simulations.

To derive mappings from the final mapping matrix, naïve descendant extraction algorithm [MS07] is used. Their idea is as such: firstly, they get maximum value in the matrix, save it into final mapping result, and then remove the row and column it located in; secondly, they continue this procedure for the left over matrix until no cell left in the matrix. This way, the number of final mappings is the minimum of number of rows and number of columns, which is the minimum of number of nodes in two graphs.

3.1.3 Network Architecture 2: Splitting Relationship Input Layer

3.1.3.1 Network Structure When two graphs have different "relationships", these relationships may also require mapping. In the previous network architecture, input layer S_{in} that represents relationships is shared between two subnets. If relationships also need to be mapped, then they can not be shared before the mapping is done.

The second network architecture shown in Figure 3.3 is proposed to account for different relationships in two graphs. Now there are two input relationship subvectors for two graphs respectively. RA_{in} represents relationships from graph A; RB_{in} represents relationships from graph B. They share one representation layer R_r. This modification enables the network to represent different relationships in two graphs, in order to map relationships other than items.

In this network, each item is represented by a single active unit in input layer (A_{in}, B_{in}) and output layer (A_{out}, B_{out}); each relationship is represented by a single active unit in relationship input layer (RA_{in}, RB_{in}). A special "self" relationship is added for the network to learn ontology graphs. The "self" relationship unit appears in both input relationship layers RA_{in} and RB_{in}.

Again, although Figure 3.3 only shows two input subvectors and two output subvectors, it's obvious that it can be generalized to multiple input subvectors and multiple output subvectors.

3.1.3.2 Training The network shown in Figure 3.3 has multiple sub networks shown in the following list. In following text, the 3-characters representing network name denote item input graph, relationship input graph, and item output graph respectively. For example ABA means subject item is from graph A, relationship is from graph B, and object item is from graph A.

The network shown in Figure 3.2 has multiple sub networks. To name a few we'll use:

1. Net_{AAA} : $\{A_{in}\text{-}AB_r\text{-}X_{AB};\ RA_{in}\text{-}R_{RA}\text{-}X_R\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_A\text{-}A_{out}$;
2. Net_{BBB} : $\{B_{in}\text{-}AB_r\text{-}X_{AB};\ RB_{in}\text{-}R_{RB}\text{-}X_R\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_B\text{-}B_{out}$;
3. Net_{AAB} : $\{A_{in}\text{-}AB_r\text{-}X_{AB};\ RA_{in}\text{-}R_{RA}\text{-}X_R\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_B\text{-}B_{out}$;
4. Net_{BBA} : $\{B_{in}\text{-}AB_r\text{-}X_{AB};\ RB_{in}\text{-}R_{RB}\text{-}X_R\ \}\text{-}H_1\text{-}W\text{-}H_2\text{-}V_A\text{-}A_{out}$;

Next, we'll use the graphs shown in Figure 3.4 as examples for describing the two types of training.

Vertical training records will be generated for each triple (subject item, relationship, object item) in each graph. Subject item will be presented at item input layer (A_{in} and B_{in}). Relationship will be represented at relationship input layer (RA_{in} and RB_{in}), and object item will be target at output layer (A_{out} and B_{out}). For example, "parent" (Relation) of "Bus" (Subject) is "Vehicle" (Object). In this example, since it's part of graph A, then

Net_{AAA} will be used. The representation of "Bus" will be activated at A_{in}, the representation of "parent" (Relation) will be activated at RA_{in}; the target at A_{out} will be the representation of "Vehicle" (Object).

Item cross training records will be generated for anchor mapping between graphs: for each triple, the object item in one graph will be replaced by mapped item in the other graph. For example, suppose one triple is in graph A : (C_{A1}, R_{a1}, C_{A2}), and one triple is in graph B: (C_{B1}, R_{b1}, C_{B2}), and there is a mapping as $m(C_{A2}, C_{B2})$. Then two cross training pairs will be generated as (C_{A1}, R_{a1}, C_{B2}), and (C_{B1}, R_{b1}, C_{A2}). Only "self" relationship is used in cross training in this dissertation. For example, "self" of "Vehicle" in ontology A corresponds to "Automobile" in Ontology B. Net_{AAB} will be used to train this record. The representation of "Vehicle" will be activated at A_{in}, the representation of "self" (Relation) will be activated at RA_{in}; the target at B_{out} will be the representation of "Automobile" (Object). Only "self" relationship is used in all cross trainings .

Network is initialized by setting the weights to small random values from a uniform distribution. The network is trained with two vertical training tasks (Net_{AAA} and Net_{BBB}), two cross training tasks (Net_{AAB} and Net_{BBA}).

One training cycle of the networks is:

1. Randomly train a record for Net_{AAA}
2. Randomly train a record for Net_{BBB}
3. Train a record for Net_{AAB} and the corresponding record for Net_{BBA} with a probability p_c

3.1.3.3 Explicit Training Method

The cross-training method described in the previous section only trains item mapping. For relationship mapping, new training method needs to be developed.

Take a look at Figure 3.3, there is a representation layer for relationships. This layer is the only connection between relationship input layers and the rest of the network. If two relationships have the same representation that is represented by the vector of activation at representation layer, then they have the same effect on the rest of the network. This fact inspires us to propose a new explicit training method.

A explicit cross training method is to train the correspondence of two relationships by directly making their representations more similar. Only a portion of the neural network is involved in this cross training method: the input subvectors and representation layer. For example, we want to train the relationship correspondence of $< R_1, R_2 >$, where R_1 belongs to graph A and R_2 belongs to graph B. R_1 will be presented at RA_{in}. The output at R_r will be recorded, which will we name as RR_1. Then R_2 is presented at RB_{in}. RR_1 will be treated as target value at R_r for R_2. Weights RU_B will be modified so that R_1 and R_1 have more similar representation at R_r with standard back propagation method. Then $< R_1, R_2 >$ will be trained so that weight RU_A will be modified to make R_1's representation at R_r similar to that of R_2. The sub networks involved in this training method will be named as $RNet_{AB}$ and $RNet_{BA}$.

With explicite training method added, training cycle in the previous section needs to be modified as following.

Network is initialized by setting the weights to small random values from a uniform distribution. The network is trained with two vertical training tasks (Net_{AAA} and Net_{BBB}), two cross training tasks (Net_{AAB} and Net_{BBA}), and two explicit training tasks ($RNet_{AB}$ and $RNet_{BA}$).

One training cycle of the networks is:

1. Randomly train a record for Net_{AAA}
2. Randomly train a record for Net_{BBB}
3. Train a record for Net_{AAB} and the corresponding record for Net_{BBA} with a probability p_c
4. Train a record for $RNet_{AB}$ and the corresponding record for $RNet_{BA}$ with a probability p_r

3.1.3.4 Testing

Item mapping is obtained by the same method as described in Section 3.1.2.3. In this section, a method is described for relationship mapping.

As mentioned in Explicit Training section, if two relationships have the same representation that is represented by the vector of activation at representation layer, then they have the same effect on the rest of the network. In this case, a metric of distance is defined between two represtation activations for two relationships.

$$d(a,b) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\frac{R_i^a - R_i^b}{2})^2}$$

where N is number of units in relationship representation layer, R_i^a is representation layer activation for relationship a, R_i^b is representation layer activation for relationship b. Because activation is between -1 and +1, difference of R_i^a and R_i^b is divided by 2 to make d between 0 and 1. This distance is just Euclidean distance normalized to [0,1], let's call it normalized Euclidean distance.

The selection of distance metrics is associated with the error function used by back propagation training. The error function used is :

$$Error = \sum_{i=1}^{N} (t_i - R_i)^2$$

where R_i is activation at node i, t_i is target at node i. This is the standard error function for back propagation. If two relationships have a small error, their effects on the rest of network are similar. By definition, their distance should also be small.

Now let's consider why cosine similarity is not used to measure similarity between two relationships. Cosine similarity is a measure of similarity between two vectors of n dimensions by finding the cosine of the angle between them; it is often used to compare documents in text mining. Cosine similarity is different from normalized Euclidean distance based similarity. Let's define normalized Euclidean distance based similarity as one minus normalized Euclidean distance so that larger values indicate greater similarity.

Two vectors could have very different cosine similarity and normalized Euclidean distance based similarity. In some cases normalized Euclidean distance based similarity is small while cosine similarity is large. One example is: the first vector is (1,1), the second vector is (0.01, 0.01). Their cosine similarity is 1 and their normalized Euclidean distance based similarity is 0.505. In some other cases normalized Euclidean distance based similarity is large while cosine similarity is small. One example is: the first vector is (0, 0.001), the second vector is (0.001, 0). Their cosine similarity is 0 and their normalized Euclidean distance based similarity is 0.9995.

Normalized Euclidean distance based similarity is a more natural measure because it is the quantity minimized by training procedure. If we use cosine similarity to test if two relationships are similar to each other, we could obtain different conclusions, so cosine similarity is not used to measure similarity of relationships.

Distance between each pair of relationship from graph A and B respectively is calculated, and a relationship distance matrix D is formed.

Similar to Section 3.1.2.3, multiple simulations with random initial weights are performed. For each simulation, a relationship distance matrix is calculated. By averaging all relationship distance matrices from all simulations, we get a final relationship distance matrix.

To derive mappings from the final relationship distance matrix, naïve descendant extraction algorithm [MS07] is used similar to Section 3.1.2.3. However, in this case, the smaller the value in matrix, the more confident the mapping is. The idea is to first obtain minimum value in the matrix, save it into final mapping result, and then remove the row and column it is located in. Then, continue this procedure for the left over matrix until no cell left in the matrix. This way, the number of final mappings is minimum of number of rows and number of columns, which is minimum of number of relationship in the two graphs.

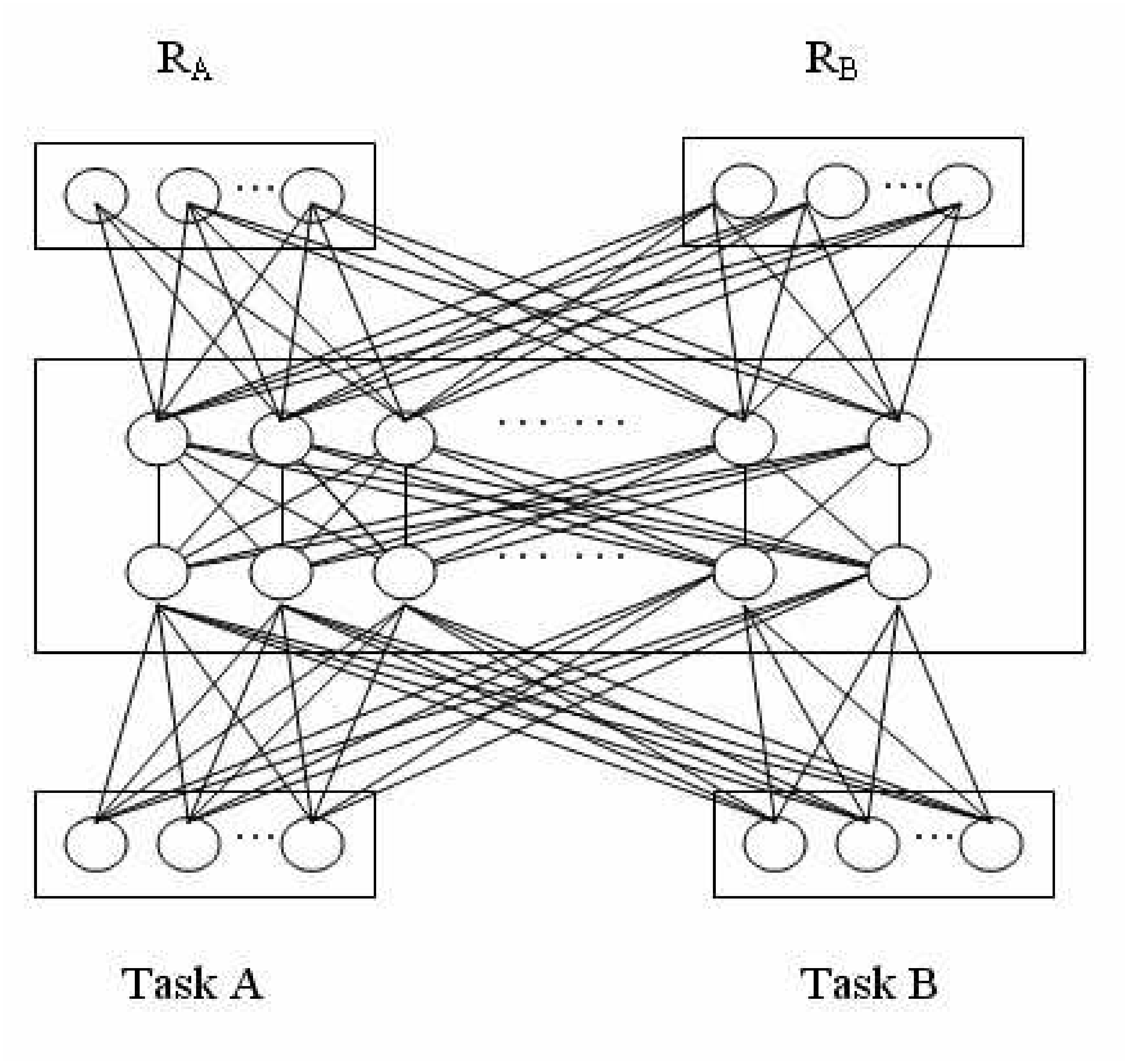


Figure 3.1: General IENN graph. Extracted from [BM06]

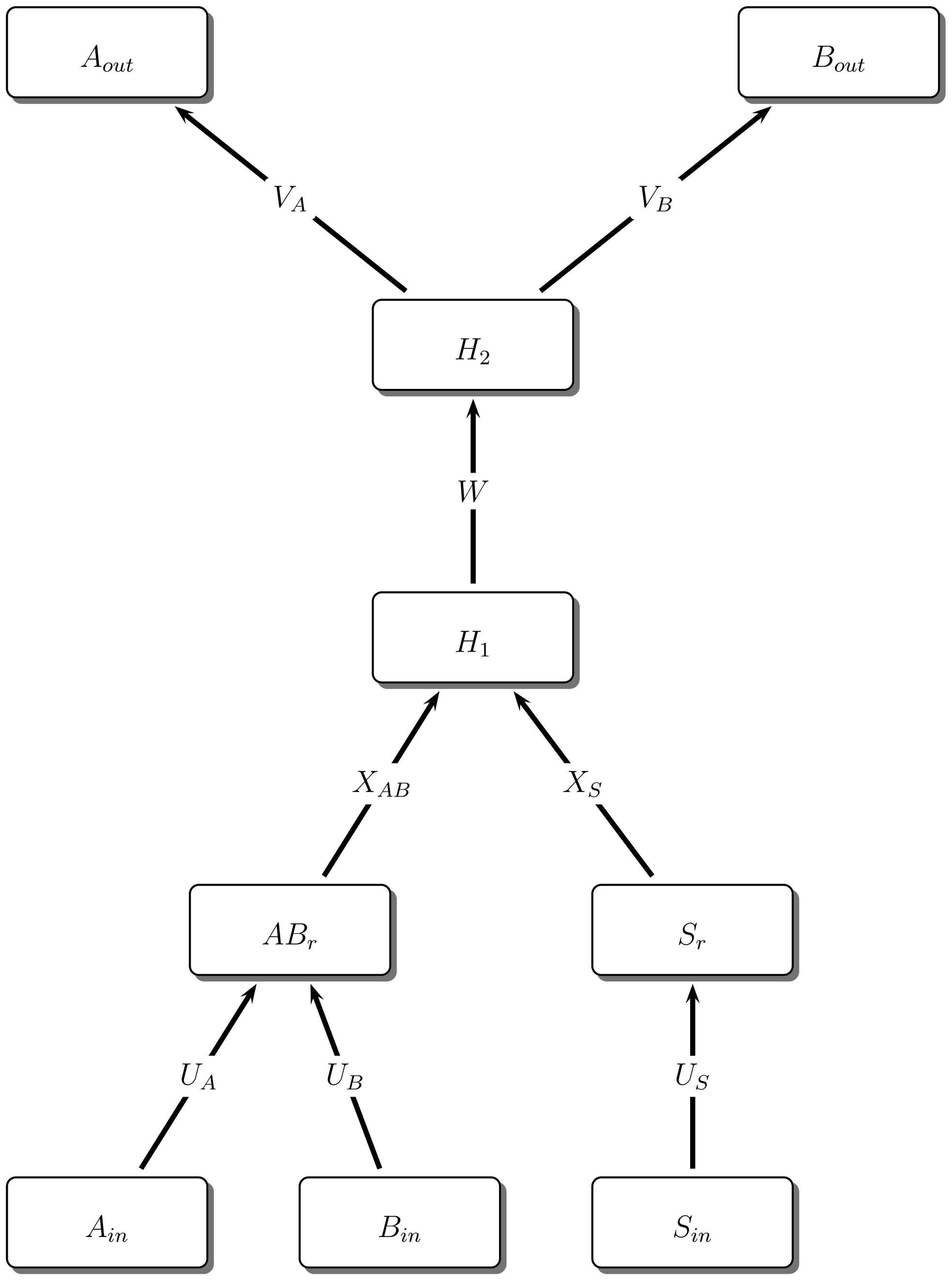


Figure 3.2: Proposed network architecture 1

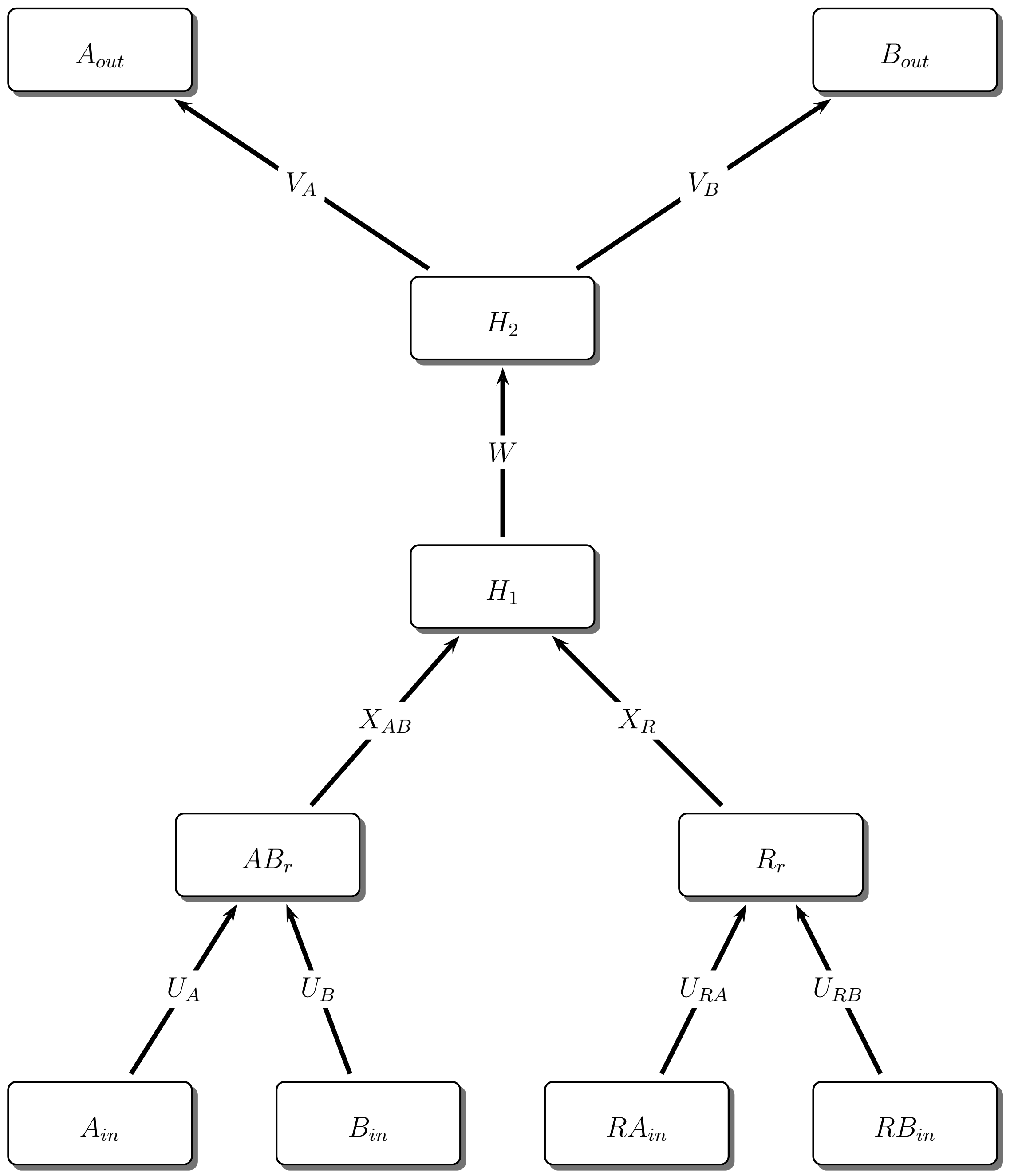


Figure 3.3: Proposed network architecture 2

3.2 APPROACH FOR ONTOLOGY MAPPING

Ontology is a formal, explicit specification of a shared conceptualization in terms of classes, properties and relations [Gru93a]. Figure 3.4 shows sample ontologies about Vehicle. The ontology at left has six classes: Object, Vehicle, Bus, Car, BMW, and Focus. The ontology at right has eight classes: Thing, Automobile, Bus, Car, Luxury Car, Family Car, BMW, and Focus. The only one relationship is "subClassOf". In this simple example, properties and/or instances are not included. Ontology mapping aims to find semantic correspondences between similar elements in two homogeneous ontologies.

In this paper, "semantic correspondence" refers to "=" relationship and the "elements" refers to classes and relationships. A mapping between similar elements e_i and e_j in O_A and O_B respectively can be written as: $m(e_i, e_j)$. In the situation of Ontology A and Ontology B, possible mappings are: $m(Object, Thing)$, $m(Vehicle, Automobile)$, $m(Bus, Bus)$, $m(Car, Car)$, $m(BMW, BMW)$, $m(Focus, Focus)$. We can also substitute the text representation of classes by character and number representation, e.g. $m(Bus, Bus)$ can be simplified to $m(B, b)$. Note that not all classes in one ontology have semantic correspondence in the other ontology, e.g. "Luxury Car" and "Family Car" in the ontology on the right side have no exact correspondence in the ontology on the left side.

In this approach, we try to extend graph mapping approach to a more generic problem: ontology mapping.

Graph mapping is only one part of ontology mapping. A successful ontology mapping approach needs to account for both syntactic and structural information [EBB+04]. In previous graph mapping approach, syntactic information is not used at all. Also because it requires training data while ontology mapping problem normally does not provide any training data, it cannot be directly used as an approach for ontology mapping.

Two problems need to be solved. First, syntactic information need to be considered, otherwise our approach won't be competitive. Second, cross training data won't be provided directly for our neural network. We either need to either generate training data first or abandon the neural network method.

To introduce syntactic information, any method that can generate textual similarity measure for each pair of mapping candidates can be used. To solve the second problem – the lack of training data – selected pairs with high textual similarity will be used as training data. Specifically, mapping pairs with similarity larger than a threshold will be used as cross training pairs. We also need to consider the similarity itself, since it conveys how confident we are about this pair. We should not treat all such pairs the same. There are two ways to use the similarity. First, we use a high threshold so that training pairs larger than or equal to this threshold will be treated as high confidence pair and other pairs with less similarity will be removed from training data. Second, we adjust learning rate of the network based on similarity. We can directly use similarity as a factor on learning rate shown as the following Equation:

$$\mu = s \times \mu_0$$

where μ_0 is original learning rate, μ is adjusted learning rate, and s is similarity.

In this dissertation, textual similarity calculation is not our focus. The simple string matching algorithm used to get textual similarity is edit distance based method. The edit distance based similarity is calculated between the name (i.e. ID) of elements based on their

Levenshtein distance. The similarity is defined by following equation, where $EditDist(i_i^A, i_j^B)$ is Levenshtein distance between elements i_i^A and i_j^B, $len(i_i^A)$ and $len(i_j^B)$ are the name length of i_i^A and i_j^B respectively.

$$NameSim(i_i^A, i_j^B) = 1 - \frac{EditDist(i_i^A, i_j^B)}{max(len(i_i^A), len(i_j^B))}$$

3.3 TEST CASES AND EVALUATION CRITERIA

3.3.1 Test Cases for graph Mapping

This study is an implementation of the 1986 family trees experiment of Hinton[Hin86]. Thus, the network architecture includes relationship units as part of the input. The original paper includes 12 relations: sister, brother, wife, husband, mother, father, uncle, aunt, son, daughter, niece, nephew. With these relationships and only 12 people in each family, there is no ambiguity, even without cross-training. Therefore, for this study, the relationships have been reduced to the following: sibling, spouse, parent, child to introduce ambiguity. The "self" relation has been added to facilitate the identification of correspondences across domains. Hinton's original study includes two family trees. For the purpose of distinguishing between the trees, English names are used in one (top tree) and Italian names are used in the other (bottom tree) as shown in Figure 3.5.

3.3.2 Test Cases for Ontology Mapping

Selected OAEI 2009[1] benchmark tests are used to evaluate OMNN approach. The domain of benchmark tests is Bibliographic references, including one reference ontology and 51 test ontologies generated from the reference ontology, while discarding selected information in order to evaluate how algorithms behave when this information is lacking. More specifically, the benchmark tests can be divided into several groups, as shown in Table 3.1, where O_R and O_T denote reference and test ontology respectively. An overview of the characteristics of each benchmark test can be found at .

All test cases share the same reference ontology, while test ontology a different. The reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. In OMNN approach, classes are treated as items, object properties and data properties are treated as relationships, and individuals are not used.

Textural information is used to generate high confident mappings, which are used as cross-training data in OMNN. However, OMNN does not focus on how well texture information is used.

In OEAI benchmark tests (Table A1), many test cases could be done perfectly with simple text based similarity, for example, in test cases 101-104, 203, 221-247, where all names and comments are the same between reference and test ontologies. If these test cases are included in evaluation, OMNN approach will get perfect mapping results simply from first step, when textual information is used to generate preliminary mappings. In OMNN approach, high confident mappings will be used as cross-training data. For these simple test

[1] http://oaei.ontologymapping.org/2009/

cases, high confident mappings are solely correct mappings. Our neural network cannot do anything to improve already perfect mappings, so these test cases are not included in this evaluation because they will not test the mapping ability of the neural network. Similarly, test cases that have the same comments or instances between reference and test ontologies highly rely on textual information to get mappings, so they are not included in evaluation; they are 201, 202, 206, 208-210, 250-252, 260, and 261. Some test cases have no structure information at all. Since our network is to use structural information, these test cases are not included in evaluation, they are 248, 253, 254, and 262. After the filtering, 19 test cases with limited texture information are selected for our experiments. They are test case 249, 257, 258, 265, 259, 266 and their sub-cases.

In all these test cases, individuals and comments do not exist in test ontology. Only class names and property names are available as textual information in test ontologies. These names are either the same as those in reference ontology or random strings. In OMNN approach, pairs having the same name are used as training pairs.

3.3.3 Evaluation Criteria

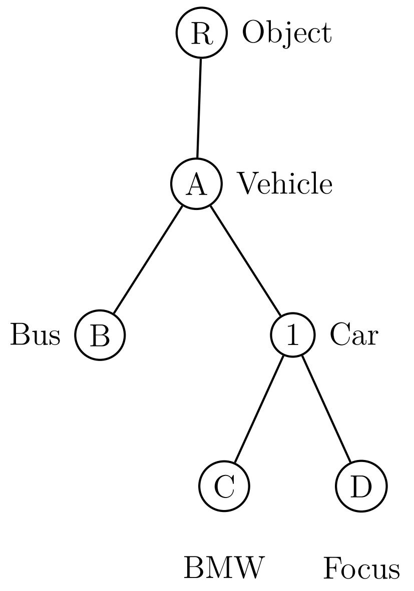
For graph mapping test cases, our focus is to evaluate whether OMNN approach can fill in missing mappings, remove ambiguities based on training data, or generate mapping results with any possible ambiguities. The evaluation criteria is primarily the number of simulations with expected results.

For ontology mapping, OAEI Benchmark tests are provided with the reference alignments. They are evaluated using standard evaluation measures *precision*, *recall* and *f-measure* computed against reference alignments. The precision, recall and f-measure are defined as follows:

$$Precision\ p = \frac{\#correct\_found\_mappings}{\#all\_found\_mappings} \tag{3.1}$$

$$Recall\ r = \frac{\#correct\_found\_mappings}{\#all\_possible\_mappings} \tag{3.2}$$

$$F - measure\ f = \frac{2 \times p \times r}{p + r} \tag{3.3}$$



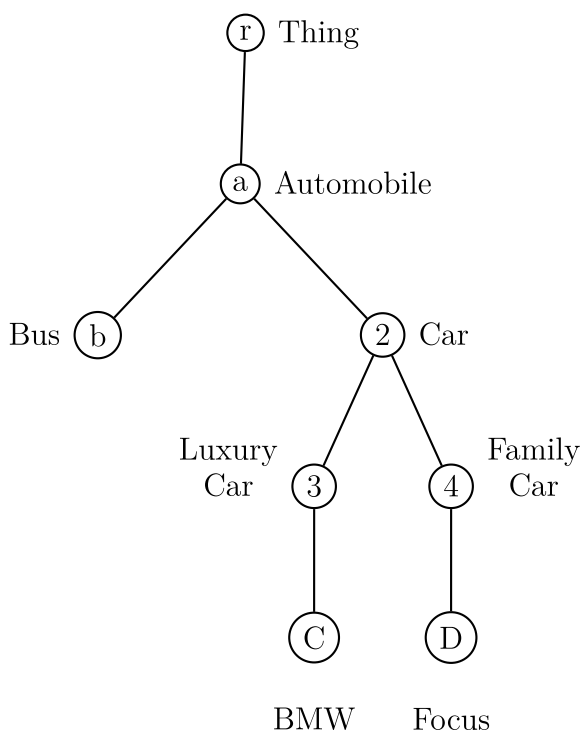
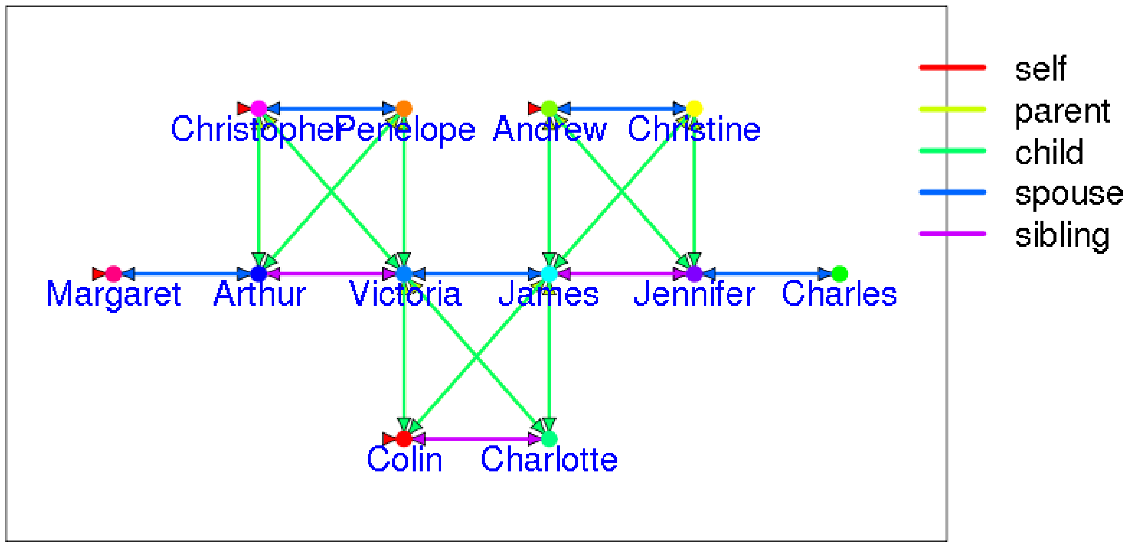


Figure 3.4: Two sample ontologies about vehicle. They can also be treated as graphs.



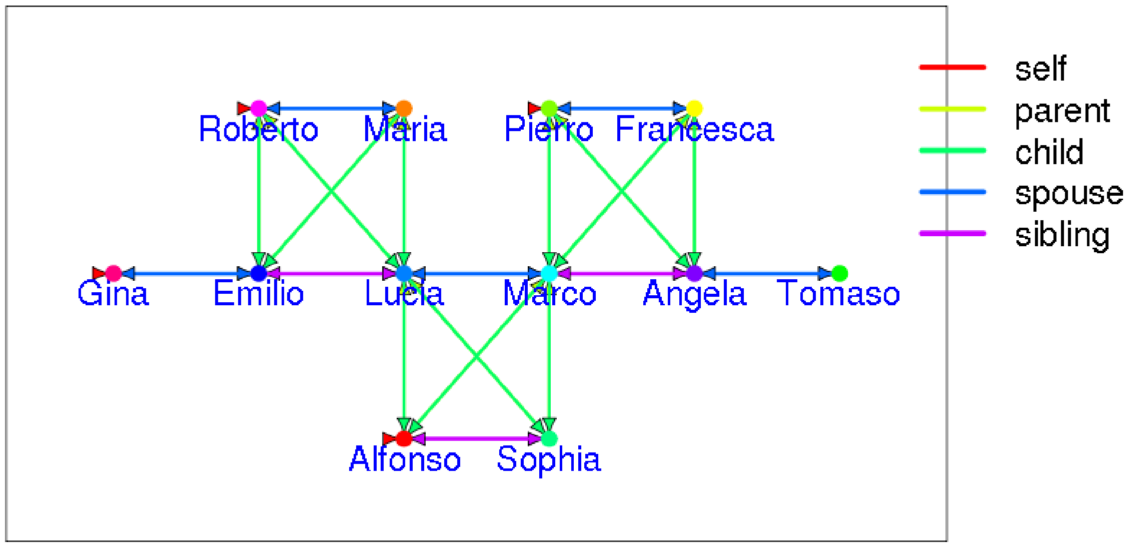


Figure 3.5: Hinton's Family Tree with Five Relationships. Note that yellow lines ("parent" relationship) are covered by green lines ("child" relationship). "Self" relationship is covered by other relationships most of the time.

Table 3.1: The description of OAEI benchmark tests

#101-104	O_R and O_T have exactly the same or totally different names
#201-210	O_R and O_T have the same graph but different linguistics in some level
#221-247	O_R and O_T have the same linguistics but different graph
#248-266	Both graph and linguistics are different between O_R and O_T
#301-304	O_T are real world cases, which we have more interest in

4.0 GRAPH MAPPING EVALUATION

Experiments are performed to obtain a deeper understanding of the OMNN architecture. Several questions to ask are:

- Can the network create node correspondence?
- Can the network create relationship correspondence?
- Can the network make simple inference?
- Is this approach scalable?
- Does it work on unequal structure mapping?

The test cases are the family tree test case shown in Figure 3.5 and several variants.

In this chapter, results of several experiments are reported. In section 4.1, four node mapping experiments are performed: the purpose is to evaluate if cross-training can help the network to disambiguate node mappings. In section 4.2, three relationship mapping experiments are performed: the purpose is to evaluate explicit training method described in 3.1.3.3. In section 4.3, the approach is tested on larger test cases with more nodes, specifically 4-layer and 5-layer family tree test cases: the purpose is to test if the OMNN approach is scalable. In section 4.4, three experiments with unequal family trees are performed: the purpose is to explore one to many node mapping (section 4.4.1) and one to many relationship mapping (section 4.4.3), as well as to test the ability to be guided to one of possible mappings in mapping task with different layers.

4.1 NODE MAPPING

In this study, four experiments are performed:

1. No cross-training
2. Cross-train (Victoria-Lucia) and (James-Marco)
3. Cross-train (Christopher-Roberto) and (Andrew-Piero)
4. Cross-train (Christine-Francesca)

10 simulations are performed for each experiment. Network with representation layer and shared relationship layer as shown in Figure 3.2 is used. Typical results for the four experiments are shown in following figures. In following figures with subplots such as Figure 4.1 and Figure 4.2, area of node is proportional to response at output layer of the network and text labels represents pattern shown in input layer. Red nodes correspond to the "correct" mapping. Because of the ambiguities, the "correct" mapping is not the only mapping that shows that the network performs as expected. In all of the following experiments, only the results from A-B are shown because cross-testing A-B and B-A generate similar results.

4.1.1 Experiment 1

The purpose of this experiment is to demonstrate that ambiguities exist in mapping result without any cross training. It's expected that cross testing will show possible ambiguity mappings. Figure 4.1 shows the variability in performance after training in Experiment 1 (no cross-training). Results were generally consistent for the vertical tasks (A-A and B-B) without ambiguity. However, performance was not always perfect – occasionally (3 in 10 simulations), the network did not converge to an optimal solution. The top figure is indicative of 6 of the 10 simulations and shows optimal performance. The middle figure is typical of 3 of the 10 simulations and accomplishes most of the task, but shows confusion on one item. In one of the 10 simulations (e.g., the bottom figure), the network does not perform well. It is expected that some fine tuning of the learning parameters will result in a more uniform convergence to optimal solutions.

While there is no ambiguity for vertical tasks, cross tasks do have ambiguities. Ambiguity could be from these possibilities:

1. Since the tree is left-right symmetric, left side of one tree could be mapped to right side of the other, and vice versa. For example, Christopher and Penelope's family could be mapped to Pierro and Francesca's family.
2. Husband and wife cannot be distinguished in cross task. For example, Christopher could be mapped to Maria, instead of Roberto.
3. Two grandchildren cannot be distinguished in cross task. For example, Colin could be mapped to Sophia, instead of Alfonso.

These ambiguity effects can also be mixed together. For top layer, effect 1 and effect 2 could be mixed, it will cause any person in top layer of the English family to be mapped to any person in top layer of Italian family. For middle layer, only reason 1 is effective, which is left-right symmetry. That will result two types of mapping: desired mapping and left-right reversed mapping. For bottom layer, there are two types of mapping: either one could be mapped to either person in the other family in the same layer.

Figure 4.2 shows the ambiguities in cross task (A-B) after training in Experiment 1 (no cross-training). The top figure shows the result with desired mapping, although it's just one of the "correct" mappings. The middle figure shows results with Christopher mapped to Maria and Penelope mapped to Roberto, which is from ambiguity reason 2; Colin is mapped to Sophia and Charlotte is mapped to Alfonso, due to the ambiguity reason 3. The bottom figure shows English family on the left side is mapped to Italian family on the right side, and vice versa, due to the ambiguity reason 1.

Figure 4.3 shows the variability of performance on the cross task (A-B) for Experiment 1. The top figure shows the average of response matrix over 10 simulations. The bottom figure shows the frequency of mapping in 10 simulations. Top layer mapping corresponds to the top left corner (4X4) of the matrix. As expected, all cells in this sub matrix have values in both figures. Middle layer mapping corresponds to the center of the matrix where the yellow "X" shape located. Two kinds of mappings dominate, i.e., one is the desired mapping (top-left to bottom-right line); the other is left-right reversed mapping (top-right to bottom-left line). The two persons in the bottom layer could be mapped to either of their correspondences in the other family, which is shown in bottom right corner (2X2) of the matrix in the figures. Although it is not evident in the figures, there are no cases of two subtrees in one family mapped to one subtree in the other family.

4.1.2 Experiment 2

In this experiment, two pairs of people in the center of the family tree are cross-trained: (Victoria-Lucia) and (James-Marco). The expected result is the ambiguity from left-right symmetry is removed.

Figure 4.4 shows the ambiguities in cross task (A-B) after training in Experiment 2. The top figure shows results: Christopher mapped to Maria and Penelope mapped to Roberto, Andrew mapped to Francesca, and Christine mapped to Pierro due to ambiguity reason 2; Colin mapped to Sophia and Charlotte mapped to Alfonso due to ambiguity reason 3. The bottom figure shows that when there is ambiguity, two nodes in one family tree could be "co-activated" by one input node in the other family tree. For example, Christopher activates both Roberto and Maria, Andrew activates both Pierro and Francesca. This is indicative of 3 of 10 of the simulations. For 8 of 10 of the simulations, middle layer is mapped correctly. In the two failed simulations, only one node in the middle layer is mapped to another node in the same layer. Correct mappings are not shown.

Figure 4.5 shows the variability of performance on the cross task (A-B) for Experiment 2. Top figure shows the average of response matrix over 10 simulations. Bottom figure shows the frequency of mapping in 10 simulations.

Top layer mapping corresponds to top-left corner (4X4) of the matrix. As expected, the network can not distinguish husband and wife, and thus Christopher and Penelope could be mapped to either Roberto or Maria, and Andrew and Christine could be mapped to either Pierro or Francesca. When comparing Figure 4.5 to Figure 4.3 (no cross-training), Christopher and Penelope can not be mapped to Pierro and Francesca in Experiment 2. Now, the English family on the left side can only be mapped to the Italian family on the left side, and the English family on the right side can only be mapped to the Italian family on the right side. This is because left-right symmetry ambiguity is removed by cross-training (Victoria-Lucia) and (James-Marco).

Middle layer mapping is reflected in the center of the matrix, where only correct mapping is left. While in Experiment 1, two kinds of mappings are indicated by a yellow “X” shape in Figure 4.3. Again, this is because of the cross-training of two pairs of nodes in the middle layer.

Same as Experiment 1, ambiguity still exists for the two persons in the bottom layer. They could be mapped to either of their correspondences in the other family, this is shown in bottom right corner (2X2) of the matrix in the figures.

4.1.3 Experiment 3

In this experiment, two pairs of people in the top of the family tree are cross-trained: (Christopher-Roberto) and (Andrew-Piero). The expected result is the ambiguity in the top layer and the middle layer is removed. For the top layer, gender ambiguity is removed because two pairs of male are cross-trained. For the middle layer, ambiguity is from left-right symmetry. The cross-trained pairs have one pair from left side and one pair from right side, hence the left-right symmetry ambiguity is also removed. For the bottom layer, the ambiguity is still there.

Figure 4.6 shows the variability of performance on the cross task (A-B) for Experiment 3. The top figure shows the average of response matrix over 10 simulations. The bottom figure shows the frequency of mapping in 10 simulations.

As expected, the network can map the two family trees perfectly except the ambiguity in the top layer and the lower layer. Same as previous experiments, ambiguity still exists for the two persons in the bottom layer. They could be mapped to either of their correspondences in the other family, which is shown in bottom right corner (2X2) of the matrix in the figures.

4.1.4 Experiment 4

In this experiment, only one pair of people in the top of the family tree are cross-trained: (Christine-Francesca). The expected result is that the ambiguity in the middle layer and the right side of top layer is removed. For the top layer, gender ambiguity is removed only in the right side of the tree because only one pair of females in right side is cross-trained. For the middle layer, ambiguity is from left-right symmetry. Even though only one pair from right side is cross-trained, if the network can map the right side to the right side, then the left side should be mapped to the left side, and hence the left-right symmetry ambiguity is also removed. For the bottom layer, the ambiguity is still there.

Figure 4.7 shows the variability of performance on the cross task (A-B) for Experiment 4. The top figure shows the average of response matrix over 10 simulations. The bottom figure shows the frequency of mapping in 10 simulations.

Top layer mapping corresponds to top-left corner (4X4) of the matrix. As expected, the network can map the two family trees perfectly except for the ambiguity in lower layer. Same as previous experiments, ambiguity still exists for the two persons in bottom layer. They could be mapped to either of their correspondences in the other family: this is shown in bottom right corner (2X2) of the matrix in the figures.

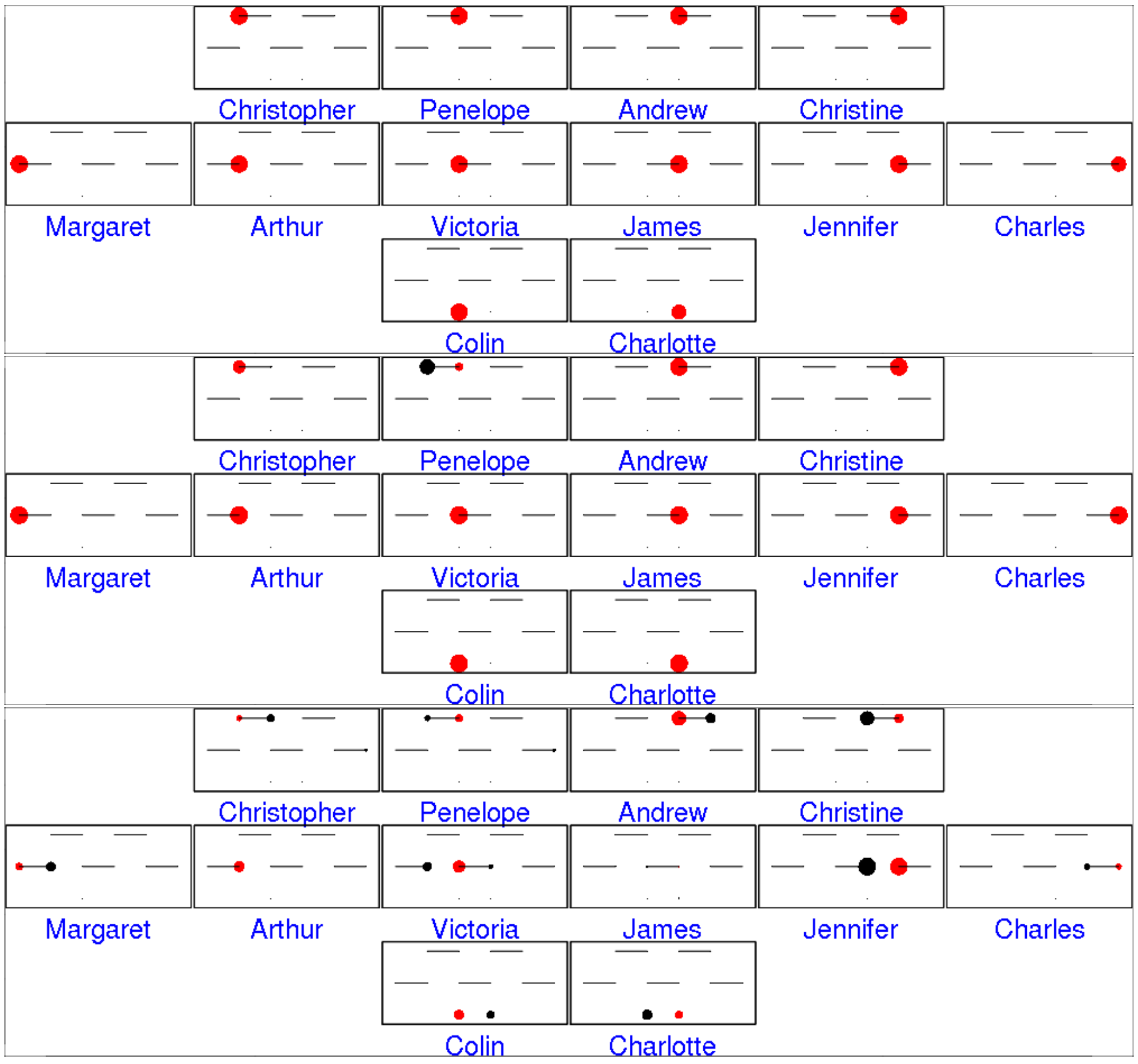


Figure 4.1: Variability or performance on the vertical identity tasks (A-A) for Experiment 1 in 4.1.1. Red nodes represent correct mapping. Top: excellent (6/10 simulations). Middle: good - outputs for Penelope is mapped to Christopher (3/10). Bottom: poor - 6 nodes are mapped wrong.

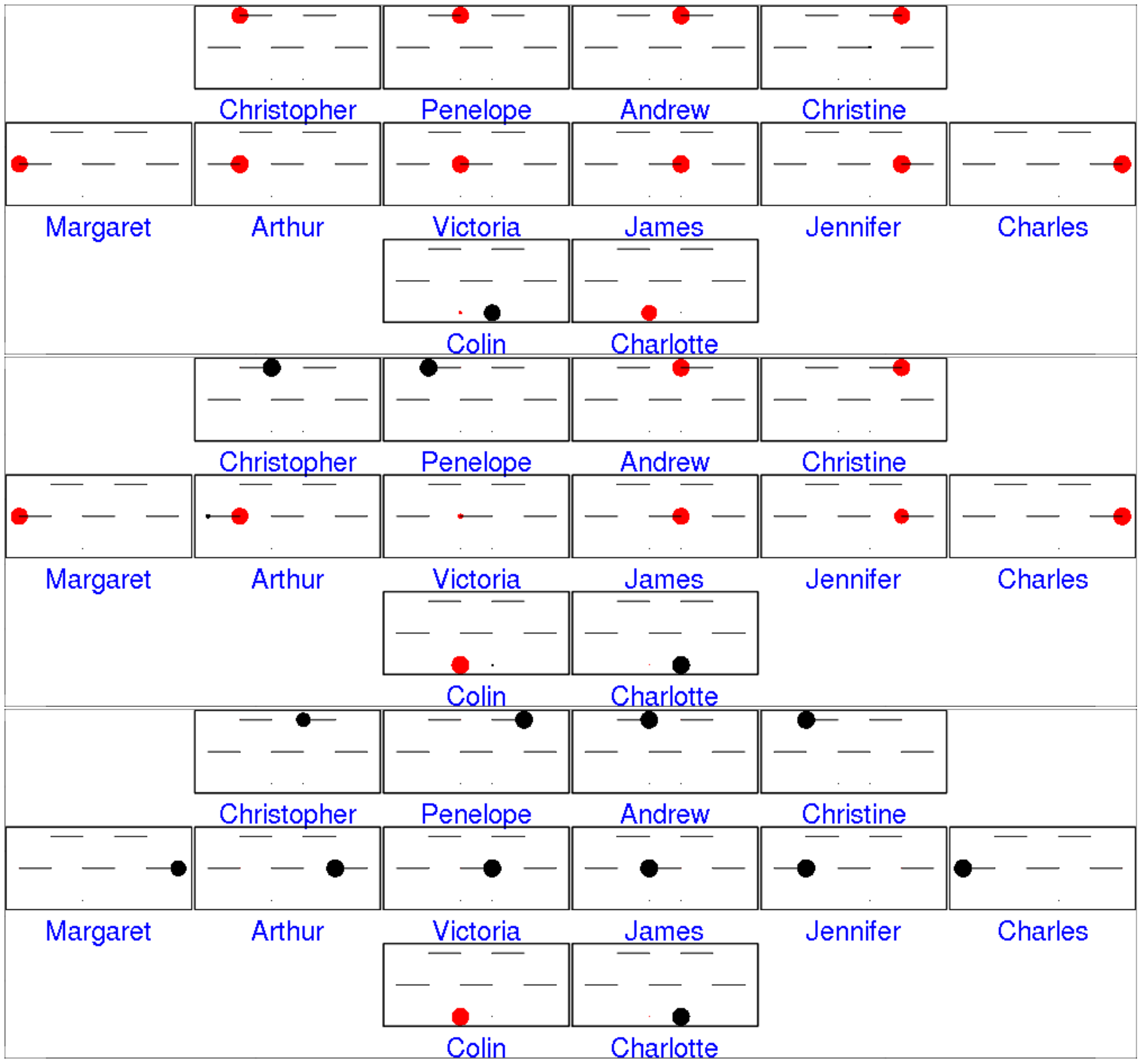
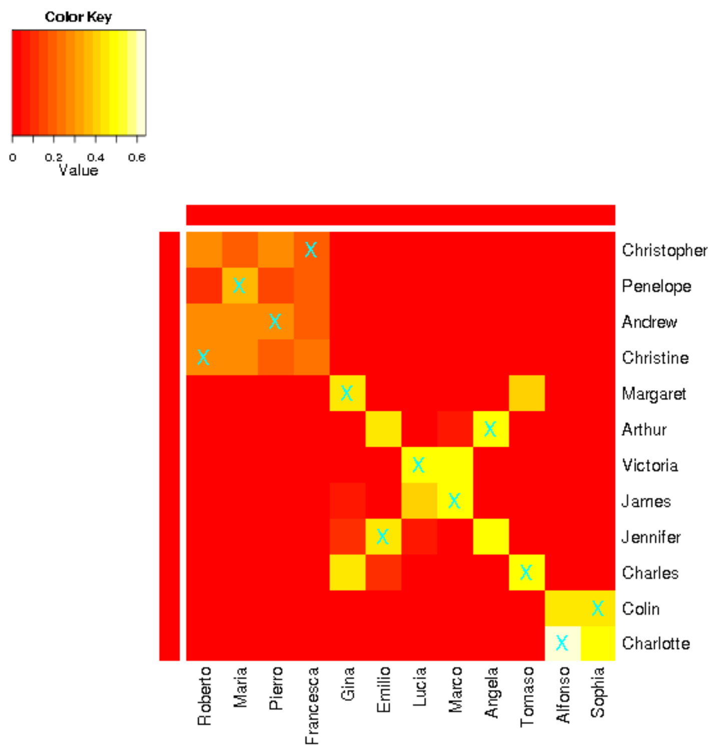


Figure 4.2: Ambiguities on the cross testing (A-B) for Experiment 1 in 4.1.1. Top: Only ambiguity is in grandchildren. Middle: Christopher and Penelope are reversed. Bottom: Left side and right side of the tree are reversed (except grand children).



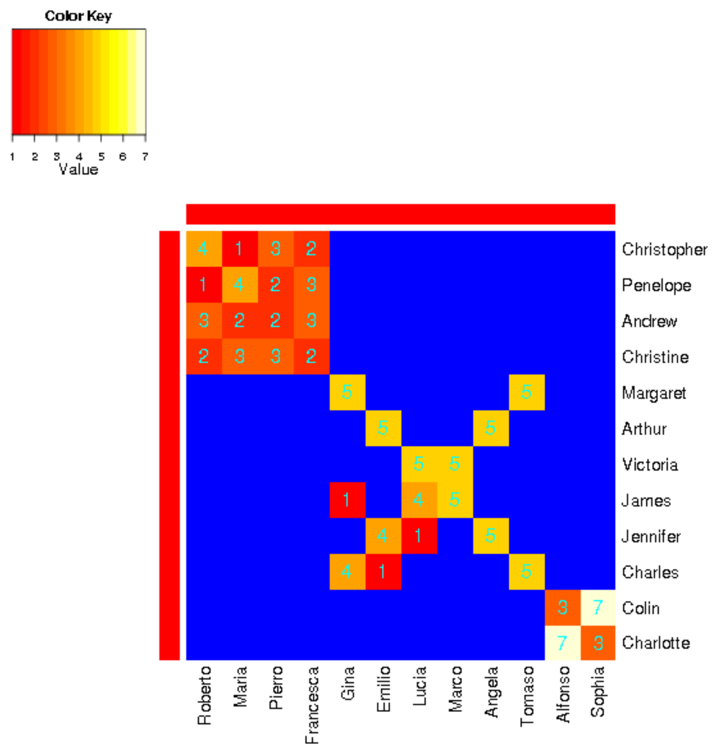


Figure 4.3: Ambiguity exists in cross testing (A-B) for Experiment 1 in 4.1.1. Top: Average response matrix after 10 simulatioins. Bottom: Mapping matrix with number of simulation in 10 simulations.

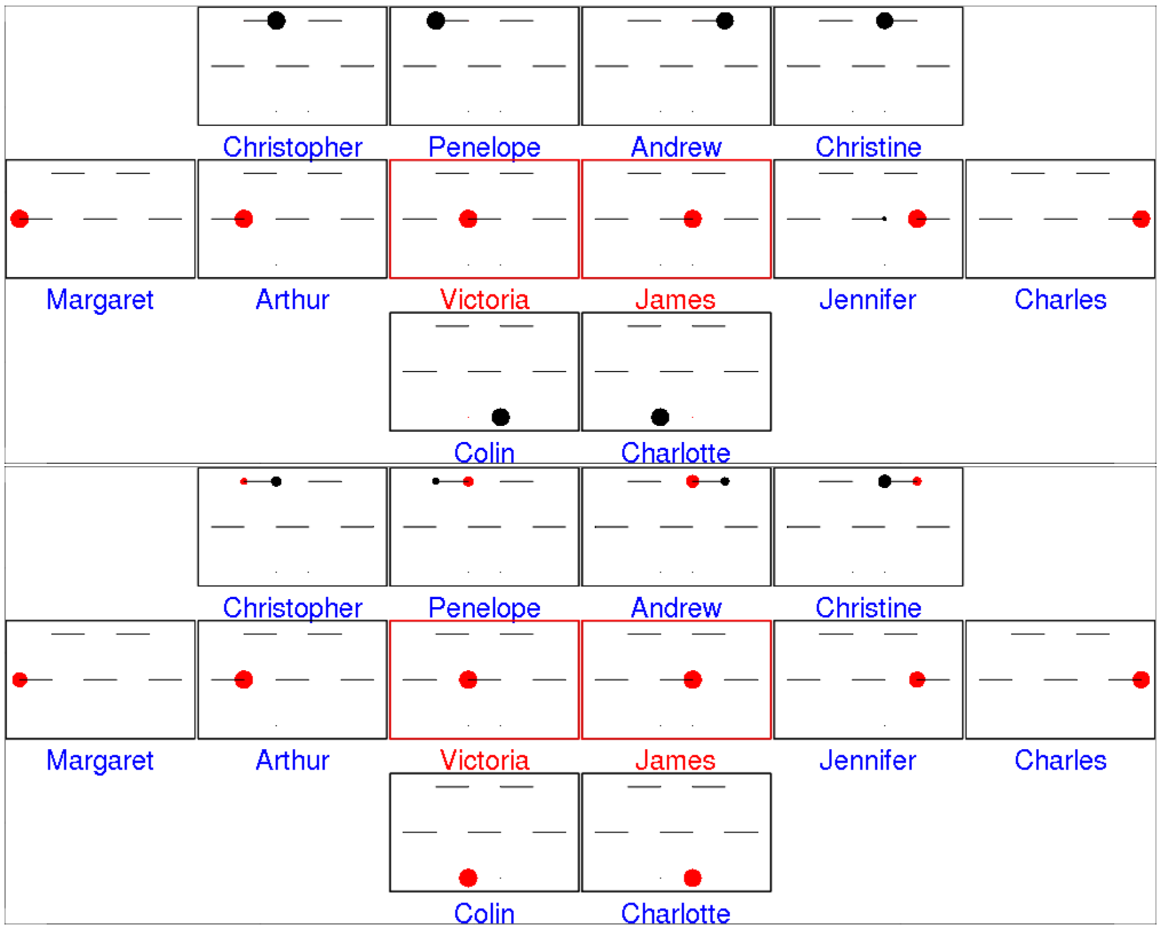


Figure 4.4: Ambiguities on the cross testing (A-B) for Experiment 2 in 4.1.2. Top: Top layer and bottom layer have ambiguity. Bottom: Christopher and Penelope are "co-activated". Correct mappings are now shown.

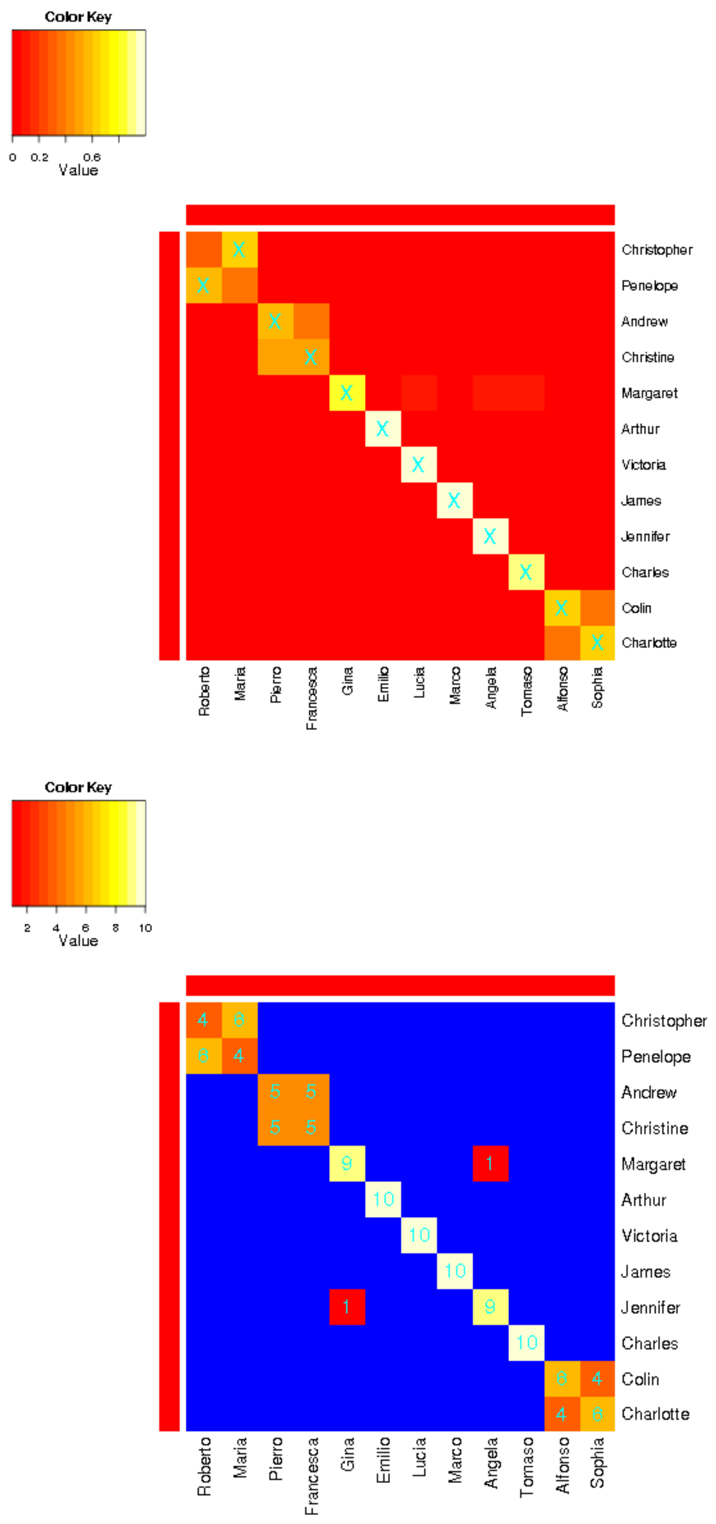
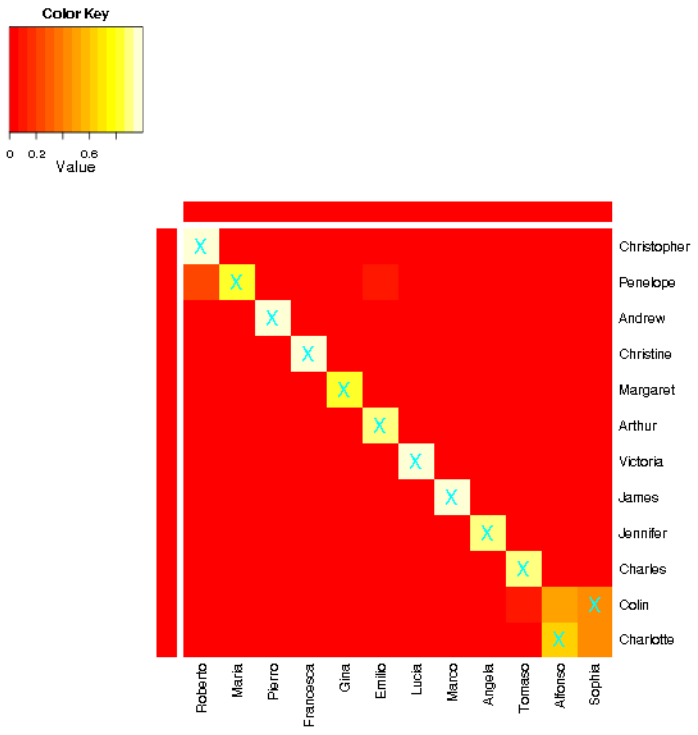


Figure 4.5: Ambiguity exists in cross testing (A-B) for Experiment 2 in 4.1.2. Top: Average response matrix after 10 simulations. Bottom: Mapping matrix with number of simulation in 10 simulations.



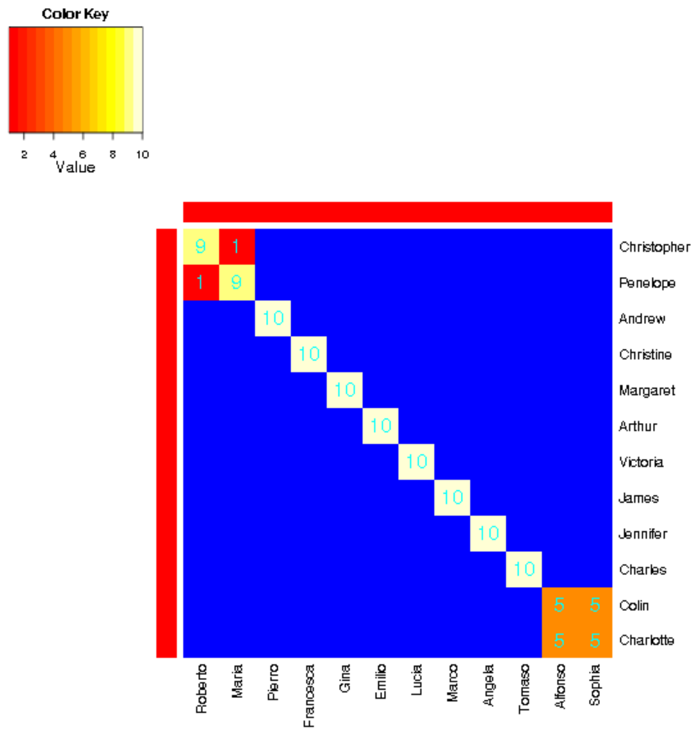
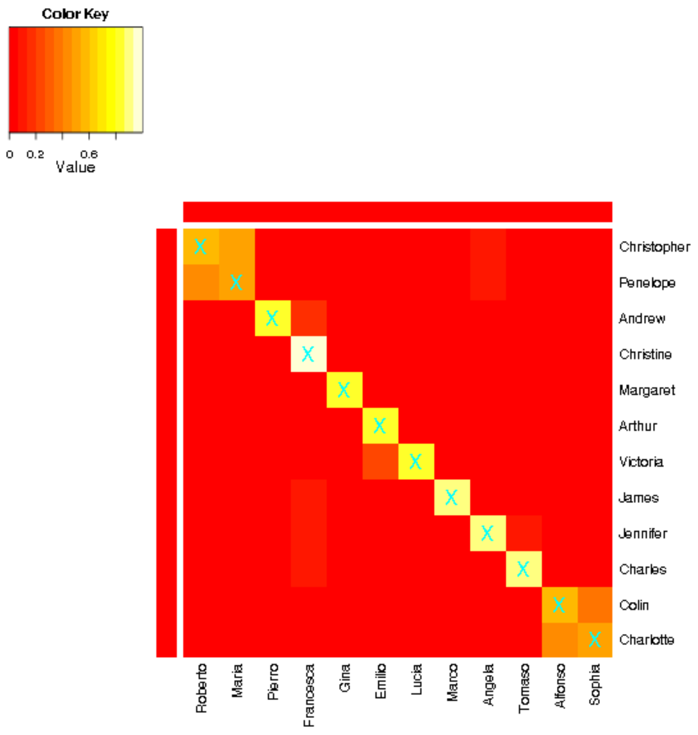


Figure 4.6: Ambiguity exists in cross testing (A-B) for Experiment 3 in 4.1.3. Top: Average response matrix after 10 simulations. Bottom: Mapping matrix with number of simulation in 10 simulations.



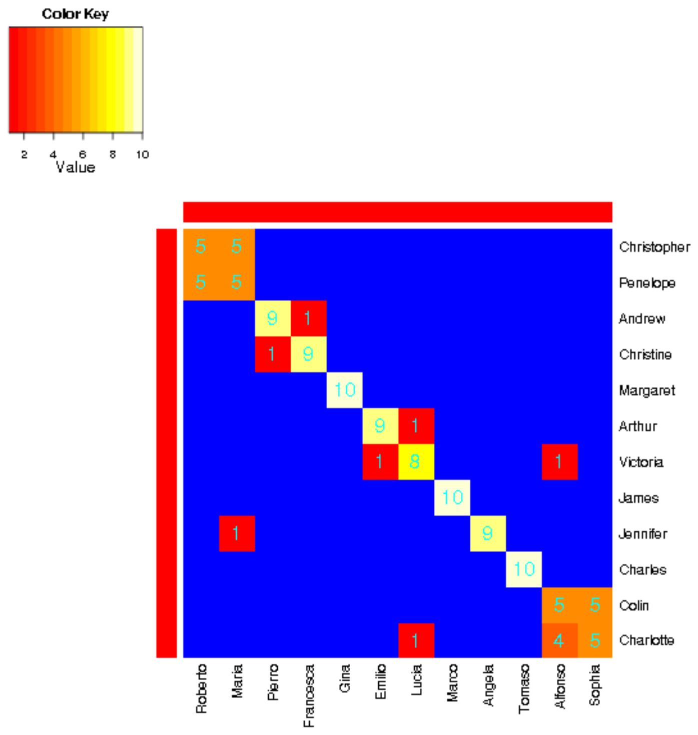


Figure 4.7: Ambiguity exists in cross testing (A-B) for Experiment 4 in 4.1.4. Top: Average response matrix after 10 simulations. Bottom: Mapping matrix with number of simulation in 10 simulations.

4.2 RELATIONSHIP MAPPING

Network with representation layer and individual relationship input layer as shown in Figure 3.3 is used for this task. Comparing this network architecture with the one in Figure 3.2, shared relationship input layer is split into two layers, one for each vertical task. The purpose is not only finding node mapping, but also finding relationship mapping. In the node mapping task, it's assumed that relationship mapping is known. In this relationship mapping task, relationship mapping is also unknown. This is a more difficult task than node mapping alone since there are more unknowns.

Three-layer family tree mapping task as described in 3.1 is used for relationship mapping experiments. Experiment 1 is vertical training only, with no cross-training or explicit relationship training. In Experiment 2, "self" relationship are explicitly trained. In Experiment 3, "self", "parent", and "spouse" relationships are explicitly trained. Performance for relationship mapping , as well as node mapping, are evaluated and compared. One to many mapping is tested in Experiment 4. In Experiment 1-3, there is no cross-training on the node.

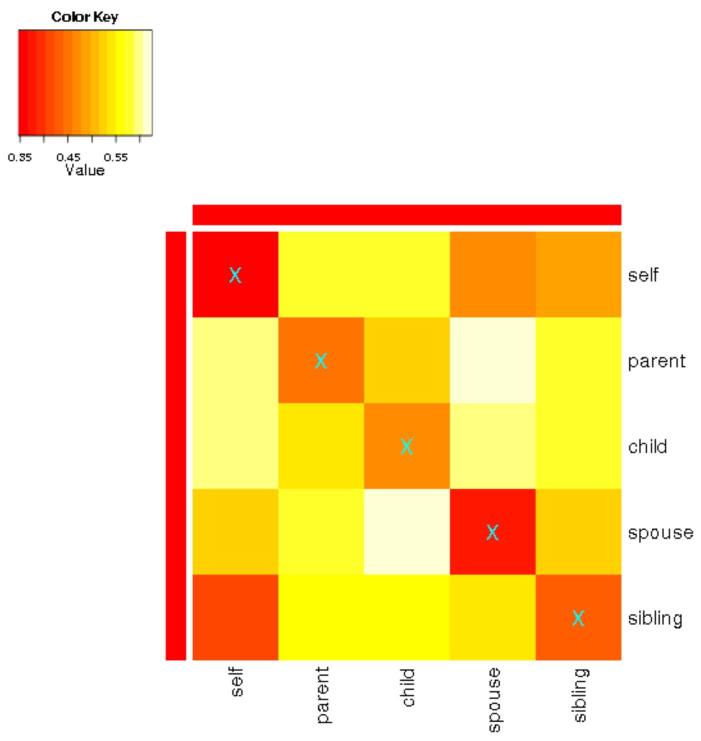
4.2.1 Experiment 1: No Explicit Training

This experiment serves as a baseline for relationship mapping without any explicit training. Average distance matrix at relationship representation layer is shown in the top of Figure 4.8. From the average distance matrix, correct mappings could be derived. Mapping matrix with simulation counts is shown in the bottom of Figure 4.8. There is still incorrect relationship mappings. Only 49 counts are falling on correct mapping diagonal line. The ratio is 49/100 =

49%.

From the simulation count matrix, relationships can be divided into two groups: “parent” and “child” in one group and“self”, “spouse” and “sibling” in another group. Mappings are mostly within one group. Only 4 of 100 cases are cross group mapping (“child” to “self” and “spouse”, “spouse” to “child”, “sibling” to “child”). One feature associated with this grouping is whether or not the relationship is cross-layer: “parent” and “child” are cross-layer relationships, while “self”, “spouse” and “sibling” are within-layer relationships. The network tends to map cross-layer relationships with cross-layer relationships, and to map within-layer relationships to within-layer relationships.

Node mapping results are shown in Figure 4.9. The top figure shows average response matrix and derived mappings. The four wrong mappings are: (Penelope, Emilio), (Christine, Maria), (Andrew, Francesca), (Arthur, Pierro). Bottom figure shows simulation count for each mapping, the sum of all numbers is $20 * 12 = 240$ and 104 counts are potential correct mappings considering ambiguity. The ratio is 43%.



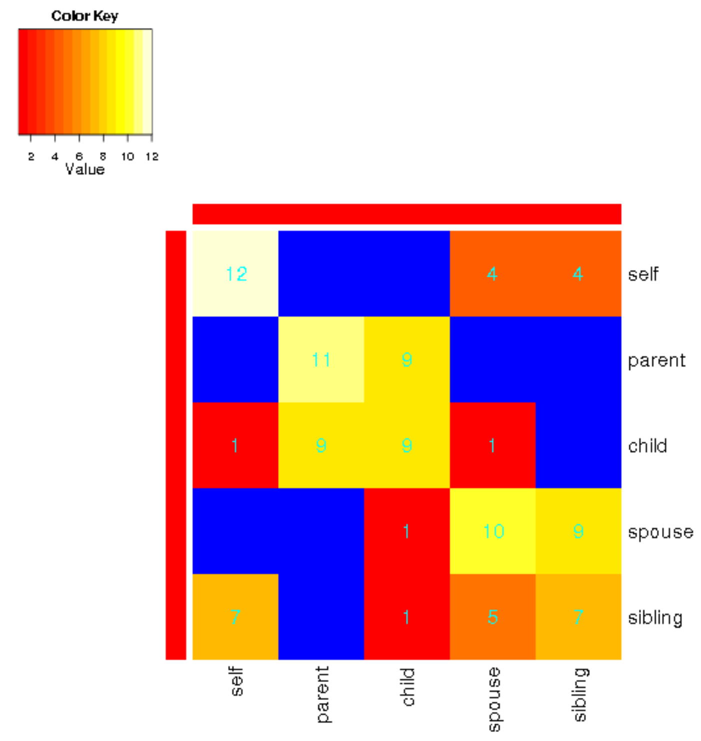
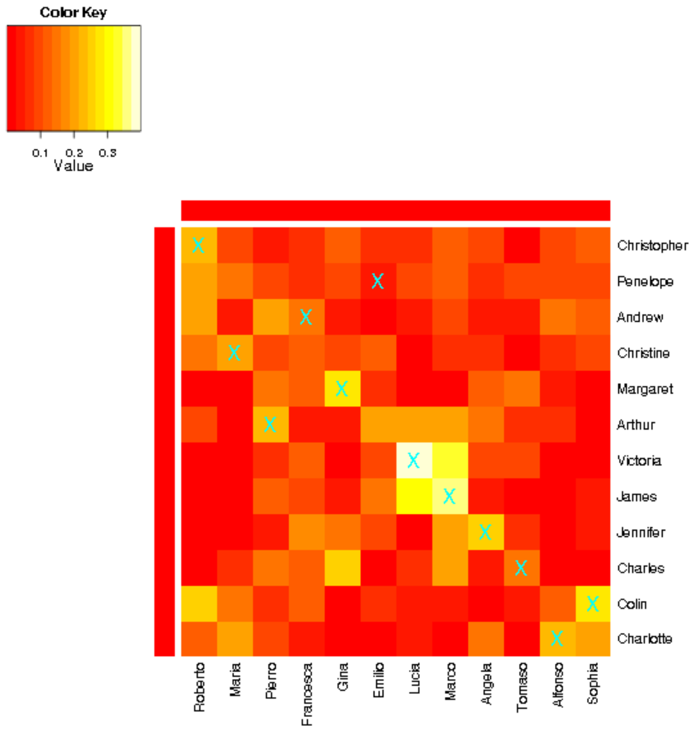


Figure 4.8: Ambiguity exists in relationship mapping (A-B) for Experiment 1 in 4.2.1. Top: Average distance matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.



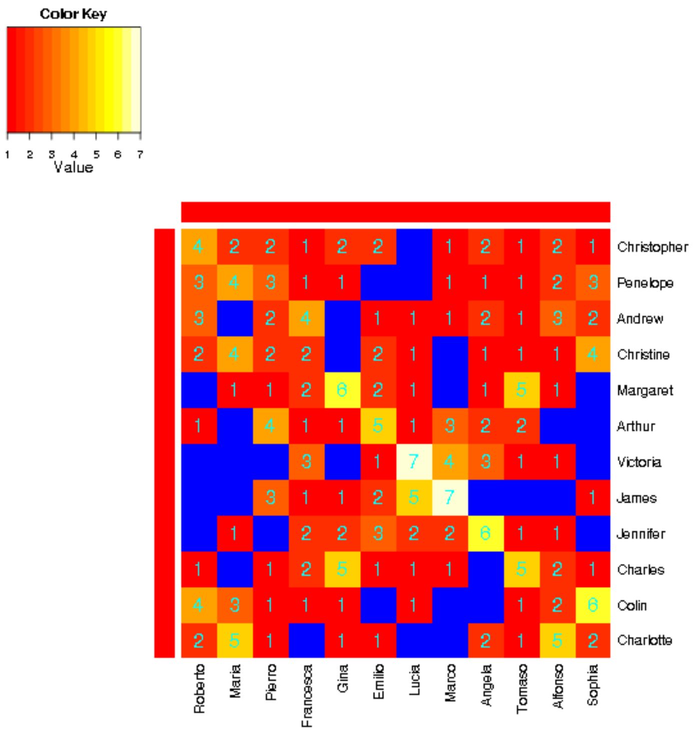


Figure 4.9: Node mapping resutls: cross testing (A-B) for Experiment 1 in 4.2.1. Top: Average response matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.

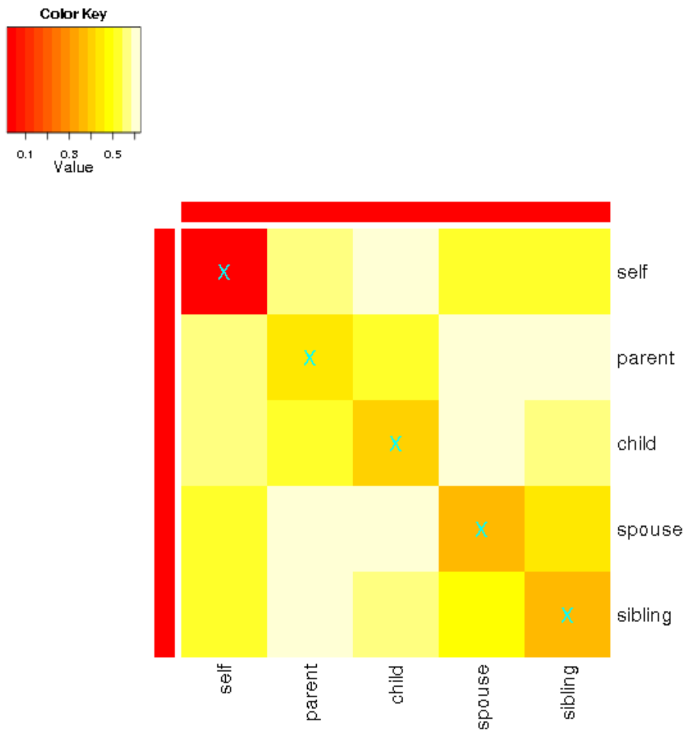
4.2.2 Experiment 2: Explicit Training on “self”

The “self” relationship is explicitly trained in this experiment. Since the “self” relationship is added to facilitate the identification of correspondences across domains, its mapping is always known.

Average distance matrix at relationship representation layer is shown in top of Figure 4.10. From the average distance matrix, correct mappings could be derived. The mapping matrix with simulation counts is shown in bottom of Figure 4.10. Some relationship mappings are still incorrect, but it is better than previous experiment in 4.2.1. 65 counts are falling on the correct mapping diagonal line. The ratio is $65/100 = 65\%$. The same ratio in Experiment 1 is 49%.

The separation of within-layer relationships and cross-layer relationships is not as apparent as in previous experiment. 8 out of 100 cases are cross group mapping.

Node mapping results are shown in Figure 4.11. The top figure shows average response matrix and derived mappings. The derived mappings are all correct when ambiguity is considered. The bottom figure shows simulation count for each mapping: the sum of all numbers is $20 * 12 = 240$, 166 counts are potential correct mappings when ambiguity is taken into consideration. The ratio is 69%. The same ratio in Experiment 1 is 43%.



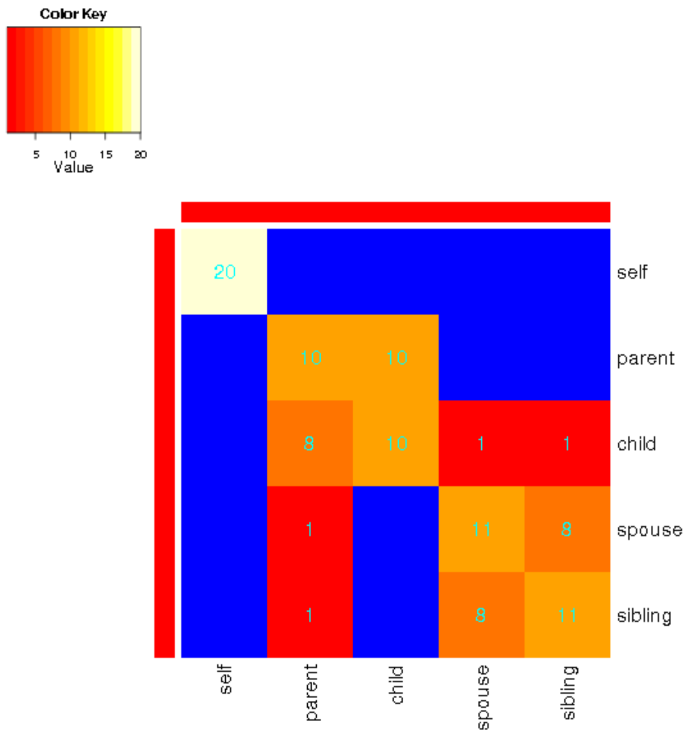
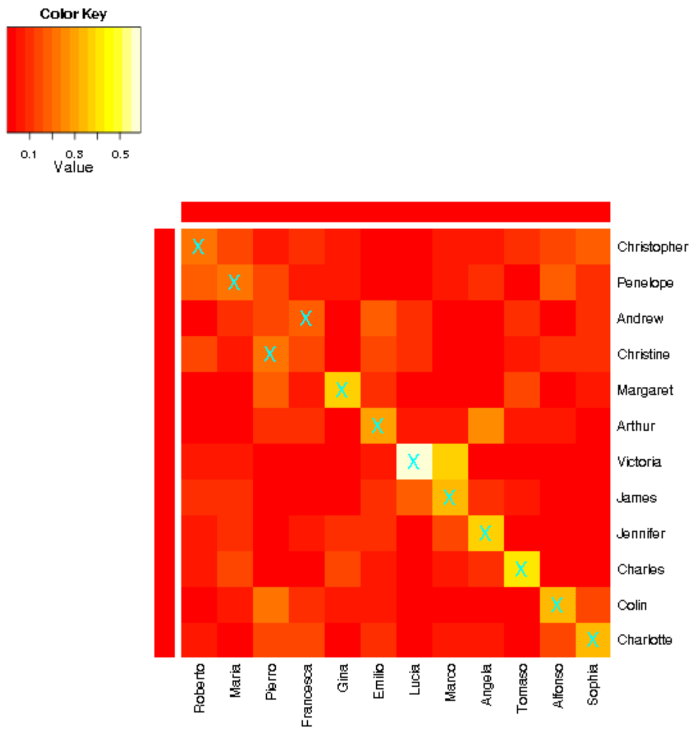


Figure 4.10: Ambiguity exists in relationship mapping (A-B) for Experiment 2 in 4.2.2. Top: Average distance matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.



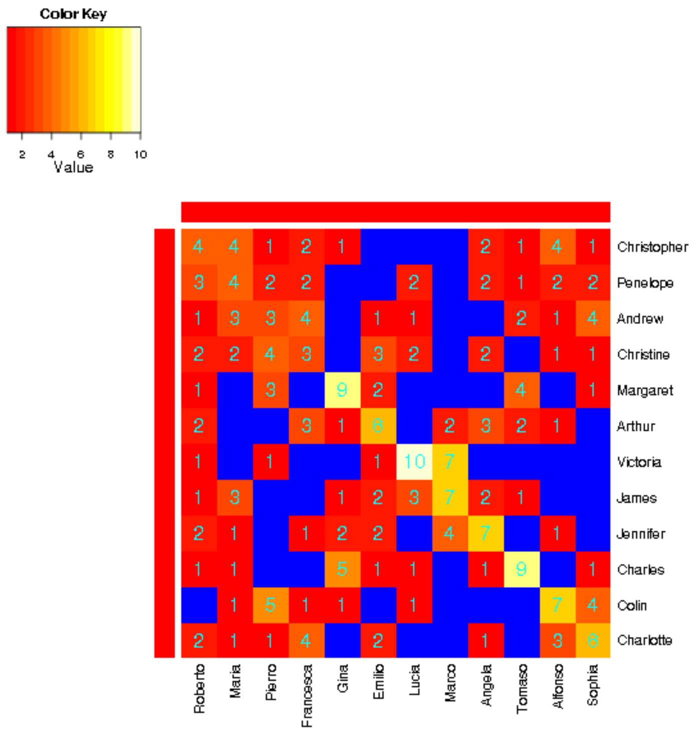


Figure 4.11: Node mapping resutls: cross testing (A-B) for Experiment 2 in 4.2.2. Top: Average response matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.

4.2.3 Experiment 3: Explicit Training on "self", "parent" and "spouse"

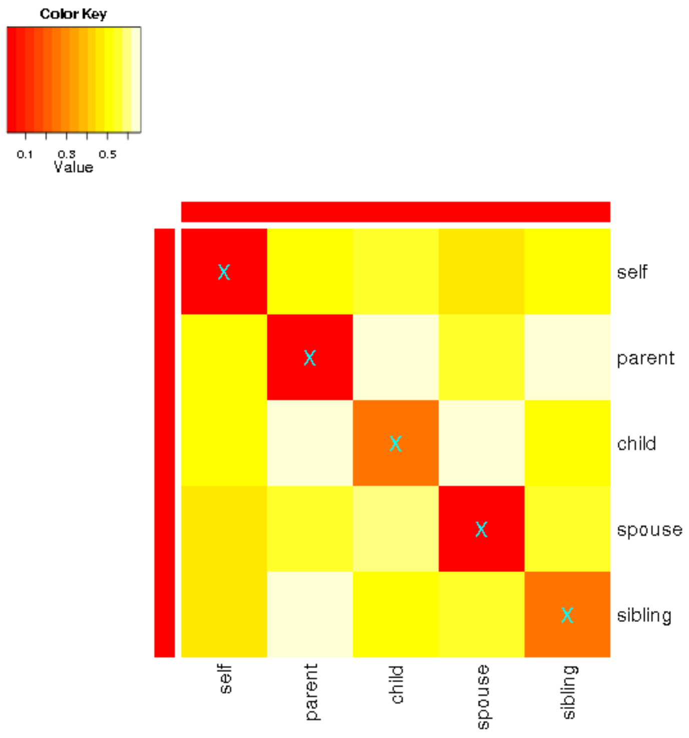
The "self", "parent" and "spouse" relationships are explicitly trained in this experiment.

Average distance matrix at relationship representation layer is shown in the top of Figure 4.12. From the average distance matrix, correct mappings could be derived. Mapping matrix with simulation counts is shown in the bottom of Figure 4.12. While some relationship mappings are still incorrect, it is better than the previous experiment in 4.2.2. 94 counts are falling on the correct mapping diagonal line. The ratio is $94/100 = 94\%$. The same ratio in Experiment 2 is 65%.

The separation of within-layer relationships and cross-layer relationships is not as apparent as in the previous experiment. 8 out of 100 cases are cross group mapping.

Node mapping results are shown in Figure 4.2.3. The top figure shows average response matrix and derived mappings. All derived mappings are correct considering ambiguity. Bottom figure shows simulation count for each mapping. The sum of all numbers is $20 * 12 = 240$, among them 181 counts are potentially correct mappings when ambiguity is taken into consideration. The ratio is 75%. The same ratio in Experiment 2 is 69%.

The segment plot for output at relationship representation layer is shown in Figure 4.14. The value is shown by the distance from the center to the radius of the segment representing the variable. Each subplot contains the result for one simulation. In each subplot, the top plot is for one network, the bottom plot is for the other. The top two plots show results where correct mapping of relationships could be found. The bottom left plot shows correct mapping could be found but not perfect. The bottom right plot shows correct mapping could not be found.



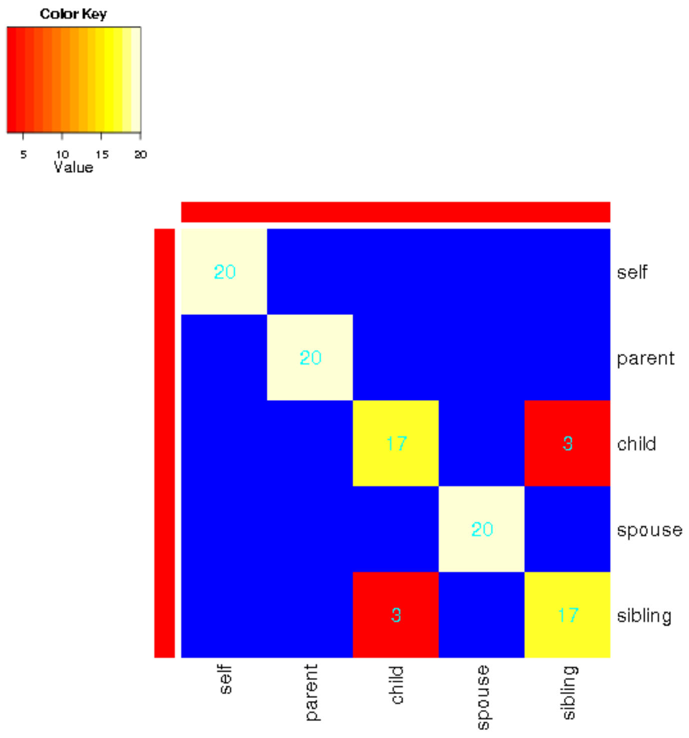
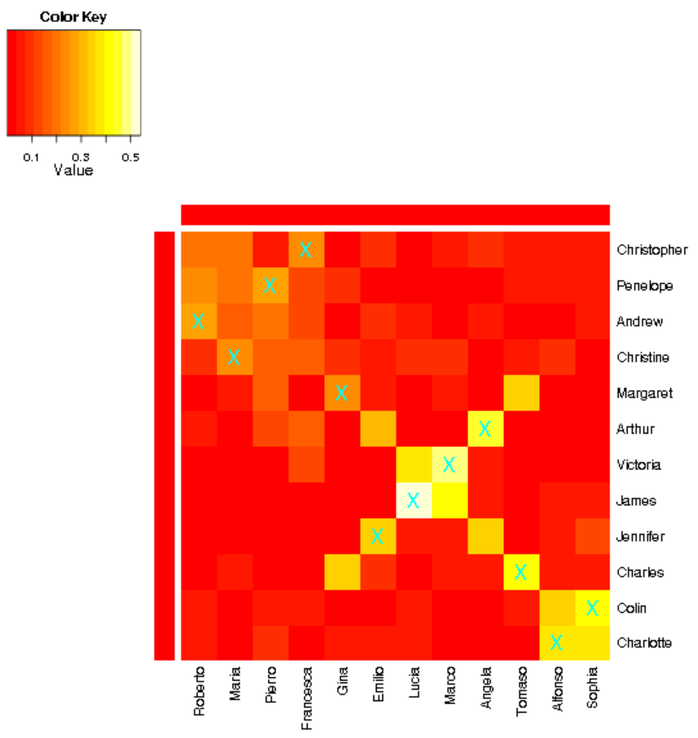


Figure 4.12: Ambiguity exists in relationship mapping (A-B) for Experiment 3 in 4.2.3. Top: Average distance matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.



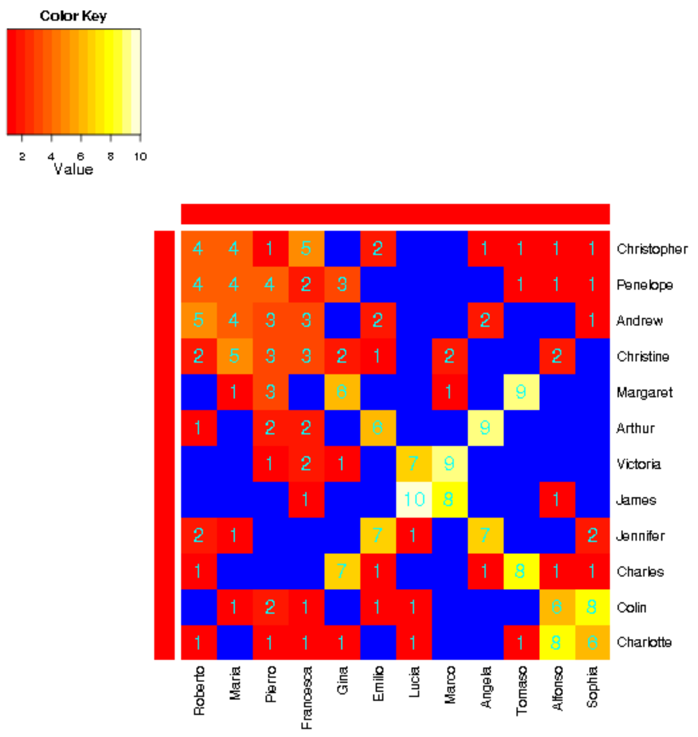


Figure 4.13: Node mapping resutls: cross testing (A-B) for Experiment 3 in 4.2.3. Top: Average response matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.

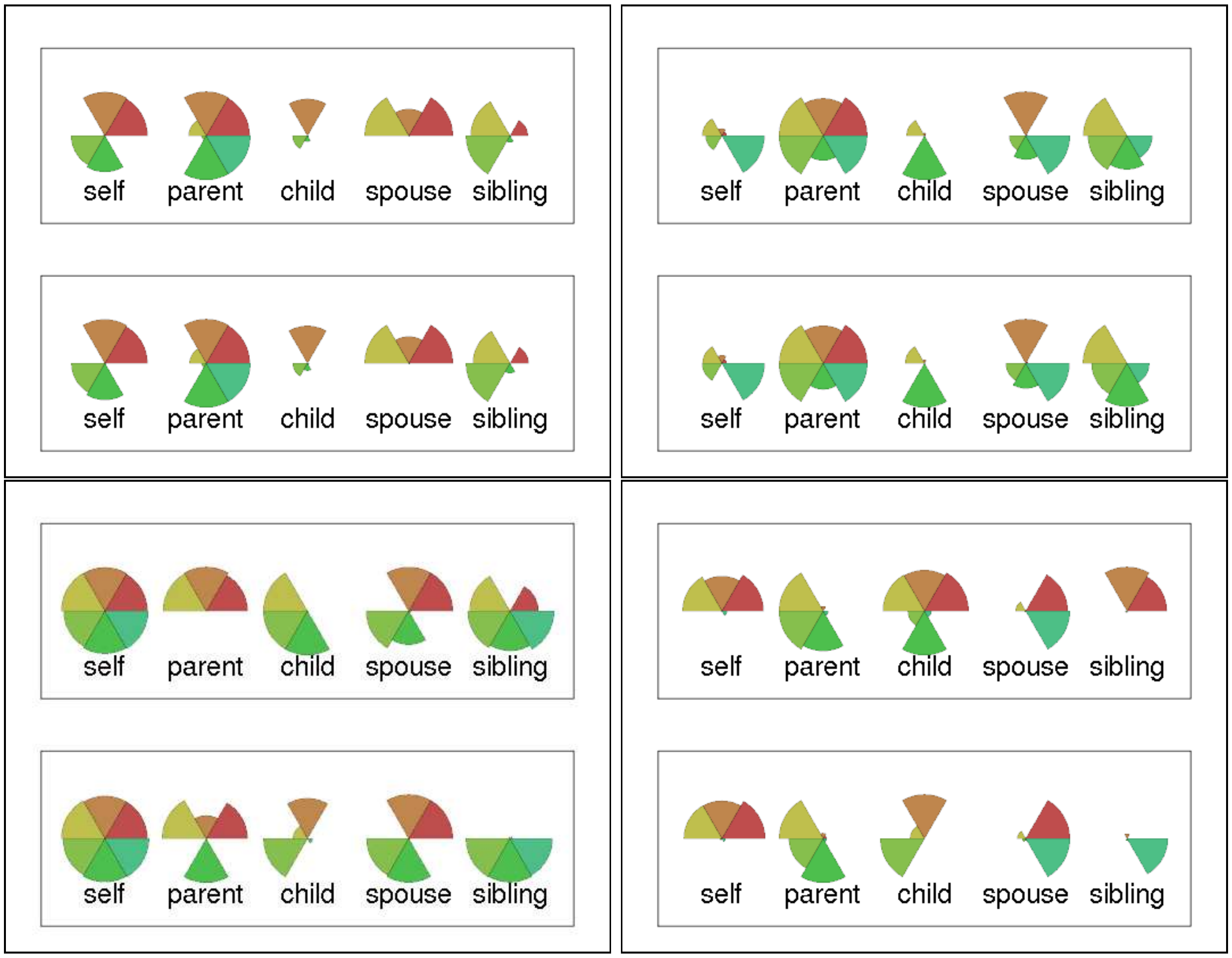


Figure 4.14: Segment plot for output at relationship representation layer for Experiment 3 in 4.2.3. The value is shown by the distance from the center to the radius of the segment representing the variable. Each subplot contains the result for one simulation. In each subplot, the top plot is for one network, the bottom plot is for the other. The top two plots show results where correct mapping of relationships could be found. Bottom left: correct mapping could be found but not perfect. Bottom right: correct mapping could not be found.

4.3 SCALABILITY TEST

To evaluate if the OMNN approach works on a larger scale problem, the family tree test case is expanded to more layers. For the 3-layer family tree as shown in Figure 3.5, 12 nodes are in each family tree. For the purpose of algorithm statement, the nodes in the 3 layer family tree are numbered from left to right, top to bottom as shown in top of Figure 4.15. To expand i layer family tree to $i+1$ layer, following algorithm is used. Suppose there are n node in i layer family tree.

1. add node (n+1) as spouse of node (n-1).
2. copy node 1 to (n+1) to node (n+2) to (2n+2), relationship is also copied.
3. connect node n and 2n with spouse relationship.
4. add node (2n+3) and (2n+4) as children of node n and 2n.

The node number of $i+1$ layer tree is $2n+4$. It could be solved that k layer family tree has $2^{k+1}-4$ nodes. 4-layer family tree has 28 nodes. 5-layer family tree has 60 nodes.

To illustrate the algorithm, example of expanding 3-layer family tree to 4-layer family tree is worked through as following. $n = 12$ in this example.

1. add node 13 as spouse of node 11.
2. copy node 1 to 13 to node 14 to 26, relationship is also copied.
3. connect node 12 and 24 with spouse relationship.
4. add node 27 and 28 as children of node 12 and 24.

The generated 4-layer family tree is shown in middle of Figure 4.15. 5-layer family tree is shown in bottom of Figure 4.15.

4.3.1 Experiment 1: 4-layer Family Tree

In this experiment, 4-layer family trees as shown in middle of Figure 4.15 are used. 28 nodes are in each family tree. The purpose of this experiment is to show that the OMNN can infer mappings correctly with cross-training. Similar to 4.1.3, in the top layer of the tree, one person from each pair of couples is cross-trained. The list of node number cross-trained is: 1;3;14;16. Four nodes are cross-trained in each family tree in total. This is about 14% of the total nodes in each tree.

The cross-training should remove all ambiguities except those at the lowest level. Since the cross-trained pairs are all at the top layer, the inference has to propagate from the top to the bottom. This experiment is more difficult than the experiment in 4.1.3 because there is one more layer for inference to propagate.

Figure 4.16 and Figure 4.17 show the variability of performance on the cross task (A-B) for Experiment 1. Figure 4.16 shows the average of response matrix over 10 simulations. Figure 4.17 shows the frequency of mapping in 10 simulations. All nodes, except two at the bottom layer are mapped correctly, with a few outliers, as shown in these two figures.

Top layer mapping corresponds to top-left corner (4X4) of the matrix. As expected, the network can map the two family trees perfectly except the ambiguity in the lower layer. Similar to the previous experiments, ambiguity still exists for the two persons in the bottom layer. They could be mapped to either of their correspondences in the other family, this is shown in bottom right corner (2X2) of the matrix in the figures.

4.3.2 Experiment 2: 5-layer Family Tree

In this experiment, 5-layer family trees, as shown at the bottom of Figure 4.15, are used. 60 nodes are in each family tree. The purpose of this experiment is to show that the OMNN can infer mappings correctly with cross-training. Similar to 4.1.3 and 4.3.1, in the top layer of the tree, one person from each pair of couples is cross-trained. The list of node number cross-trained is: 1;3;14;16;30;32;43;45. 8 nodes are cross-trained in each family tree in total. This is about 13% of total nodes in each tree.

The cross-training should remove all ambiguities except at lowest level. Since the cross-trained pairs are all at the top layer, the inference has to propagated from top to bottom. This experiment is more difficult than the experiment in 4.1.3, because there is one more layer for inference to propagate.

Figure 4.18 and Figure 4.19 show the variability of performance on the cross task (A-B) for Experiment 2. Figure 4.18 shows the average of response matrix over 10 simulations. Figure 4.19 shows the frequency of mapping in 10 simulations. All nodes except the two at the bottom layer are mapped correctly, with several outliers, as shown in these two figures.

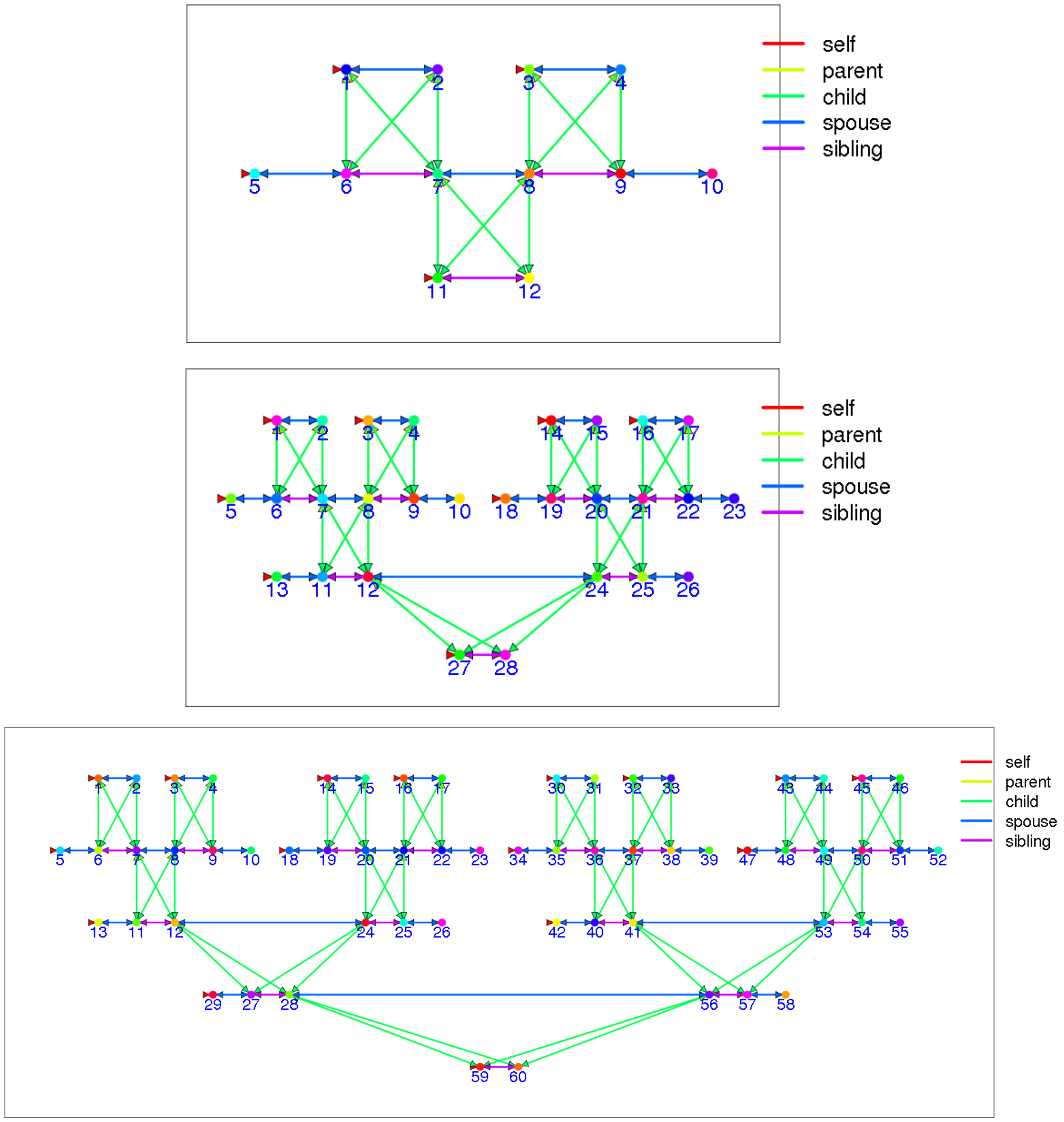


Figure 4.15: Family tree test cases. Top: 3 layers. Middle: 4 layers. Bottom: 5 layers. Note that the yellow lines (“parent” relationship) are covered by the green lines (“child” relationship). “Self” relationship is covered by other relationships most of the time.

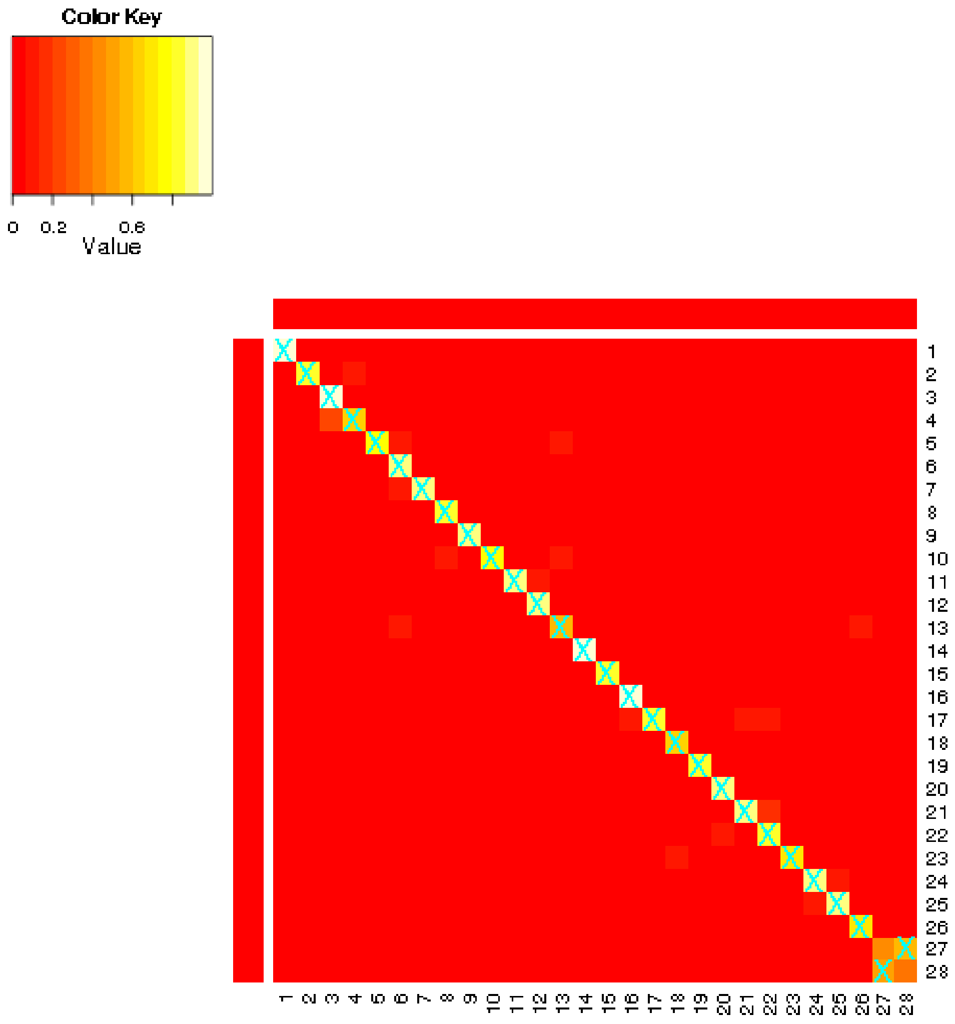


Figure 4.16: Ambiguity exists in cross testing (A-B) for Experiment 1 in 4.3.1. Average response matrix after 10 simulations.

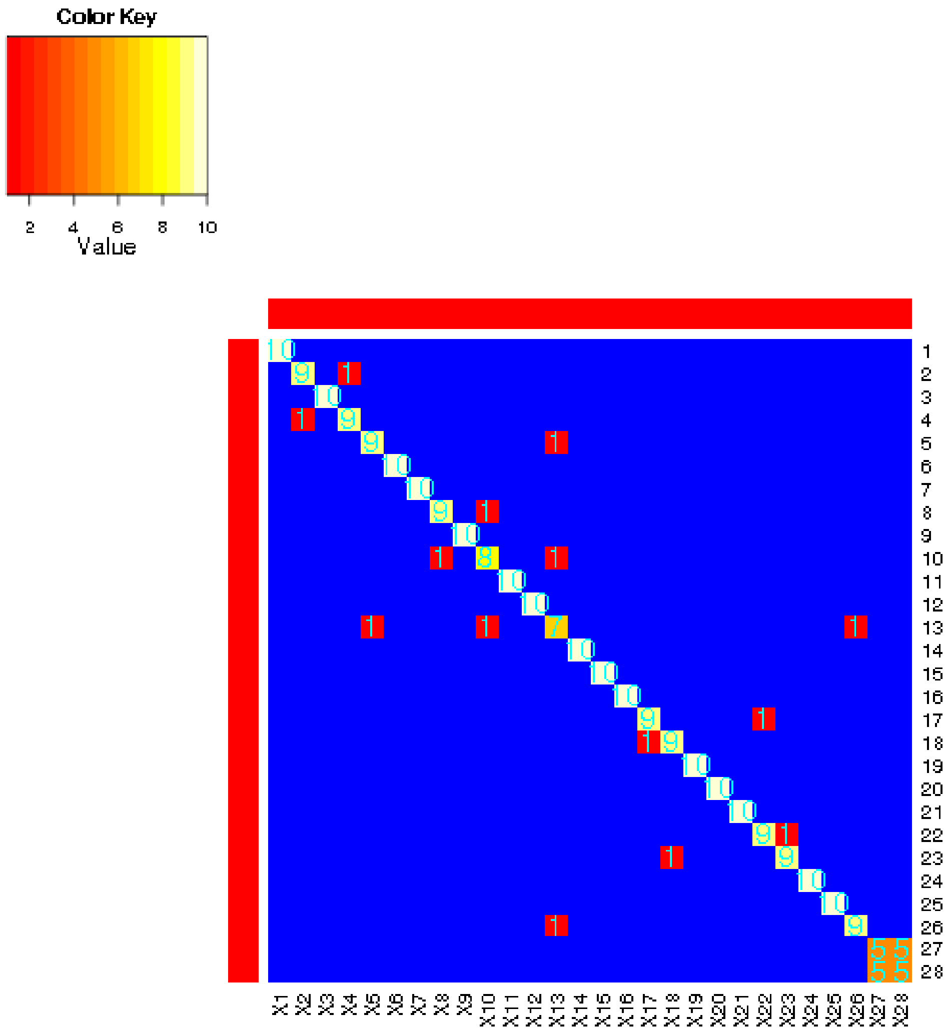


Figure 4.17: Ambiguity exists in cross testing (A-B) for Experiment 1 in 4.3.1. Mapping matrix with number of simulation in 10 simulations.

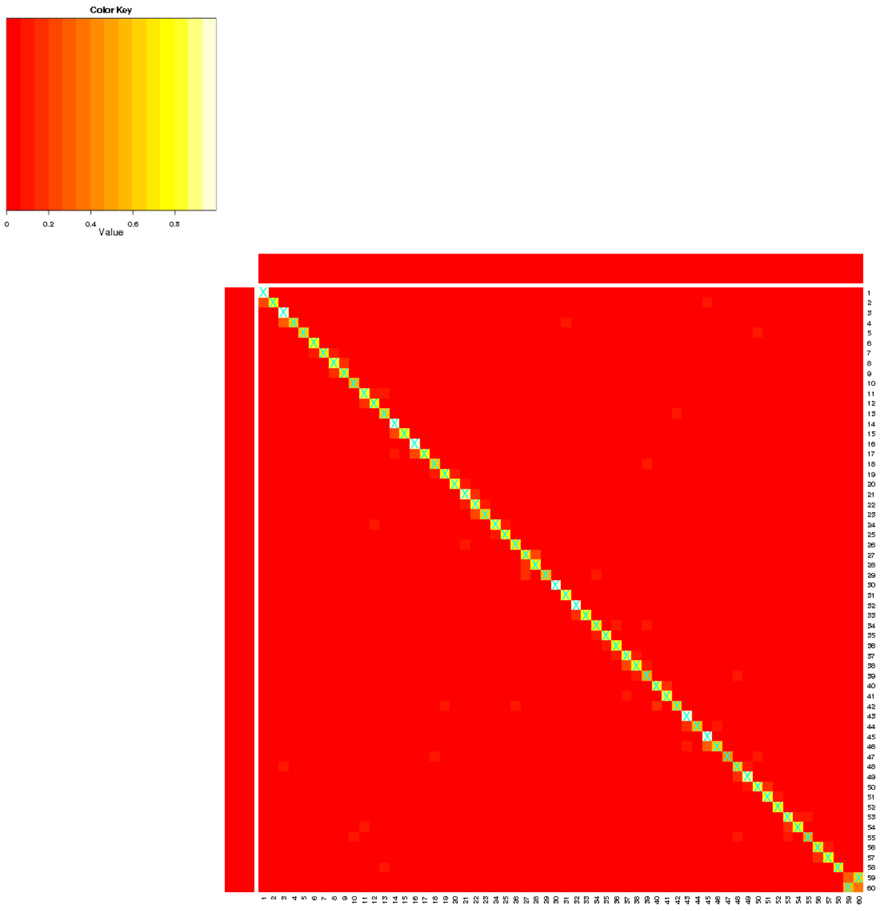


Figure 4.18: Ambiguity exists in cross testing (A-B) for Experiment 2 in 4.3.2. Average response matrix after 10 simulations.

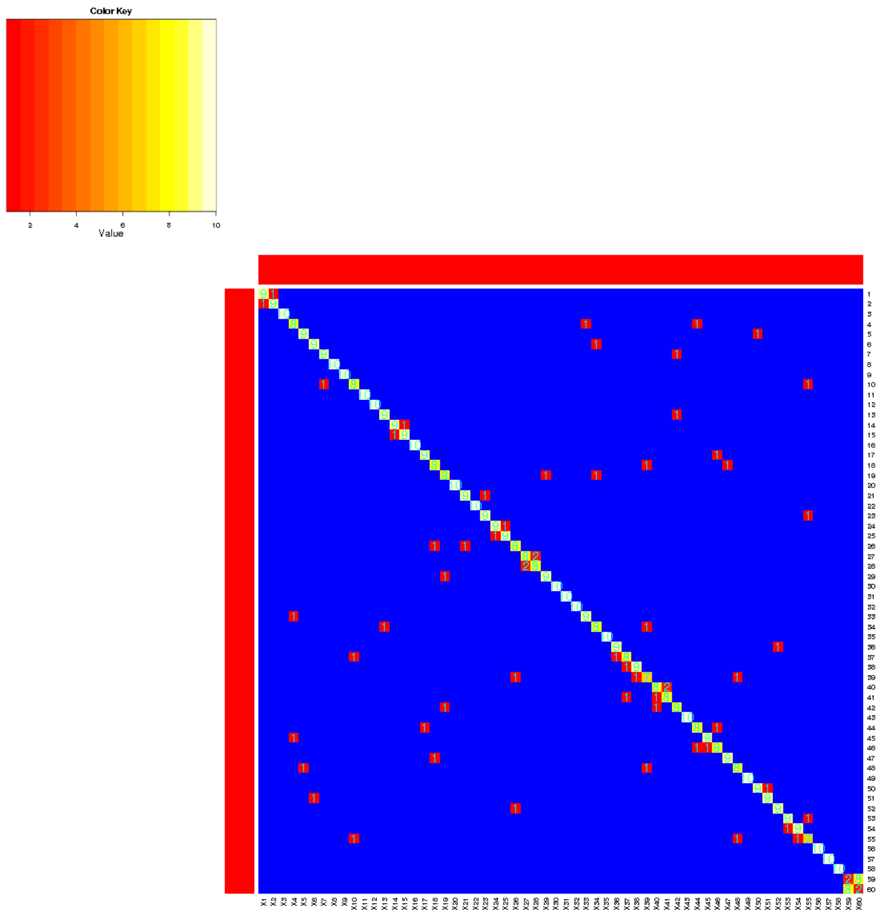


Figure 4.19: Ambiguity exists in cross testing (A-B) for Experiment 2 in 4.3.2. Mapping matrix with number of simulation in 10 simulations.

4.4 UNEQUAL GRAPH MAPPING

4.4.1 Experiment 1: Different Number of Nodes

In this experiment, two 3-layer familytrees are used. They are similar to those shown in Figure 3.5, except in the Italian family, “Sophia” is removed.

Just as experiment setting in 4.1.3, two pairs of people in the top of the family tree are cross-trained: (Christopher-Roberto) and (Andrew-Piero).

Network with representation layer and shared relationship layer as shown in Figure 3.2 is used. 10 simulations were performed.

Figure 4.20 shows the variability of performance on the cross task (A-B) for Experiment 3. The top figure shows the average of response matrix over 10 simulations. The bottom figure shows the frequency of mapping in 10 simulations. As expected, the network can map the two family trees perfectly. “Colin” and “Charlotte” are both mapped to “Alfonso” with roughly the same times of simulation, and the average responses are equally strong.

4.4.2 Experiment 2: 3-Layer to 4-Layer Mapping

In this experiment, two familytrees with different layers are mapped. Specifically, 3-layer familytree is mapped to 4-layer family tree. The two trees are shown in Figure 4.15.

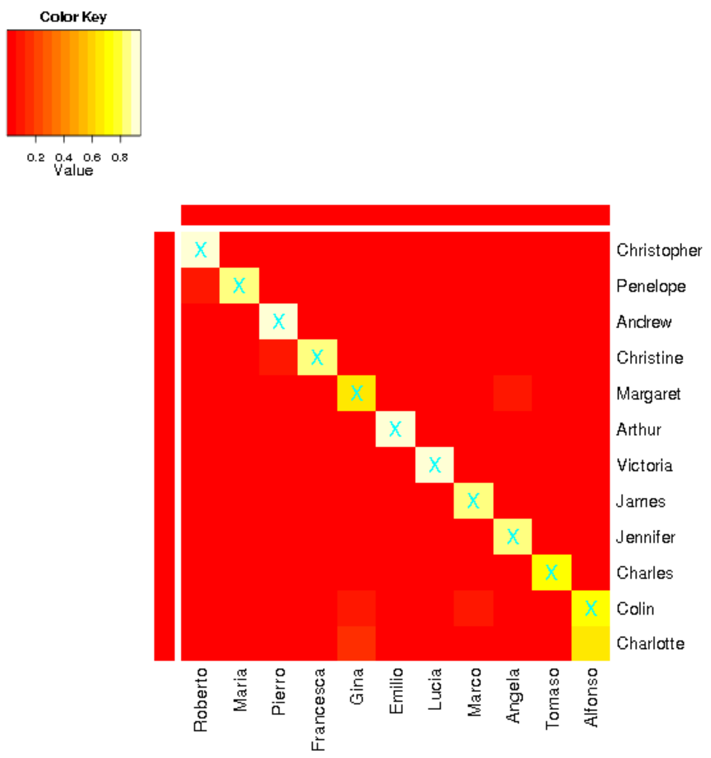
There are at least 3 ways to map the 3-layer familytree to 4-layer family tree. The first way is to map 3-layer family tree to nodes 1-12 (a subtree) in 4-layer familytree. The second way is to map 3-layer family tree to nodes 14-25(another subtree) in 4-layer familytree. The third way is to map nodes 1-12 in 3-layer familytree to nodes 7, 8, 20, 21, 13, 11, 12, 24, 25, 26, 27 28 in 4-layer familytree.

In the first test case, first way of mapping is the desired mapping. Nodes 1-4, 11, 12 in 3-layer family tree are cross-trained. 20 simulations were performed. Network with representation layer and shared relationship layer, as shown in Figure 3.2, is used.

Figure 4.4.2 shows average response matrix and derived mappings. The derived mappings are all correct.

In the second test case, the third way of mapping is the desired mapping. Nodes 1-4, 11, 12 in 3-layer familytree are cross-trained. 20 simulations were performed. Network with representation layer and shared relationship layer as shown in Figure 3.2 is used.

Figure 4.4.2 shows average response matrix and derived mappings. The derived mappings are all correct.



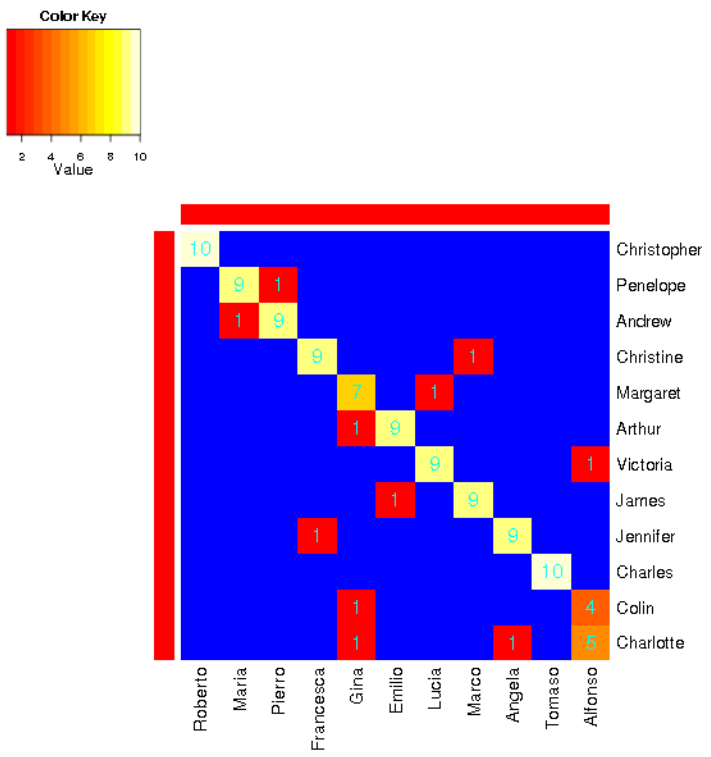


Figure 4.20: Cross testing (A-B) for Experiment in 4.4.1. Top: Average response matrix after 10 simulations. Bottom: Mapping matrix with number of simulation in 10 simulations.

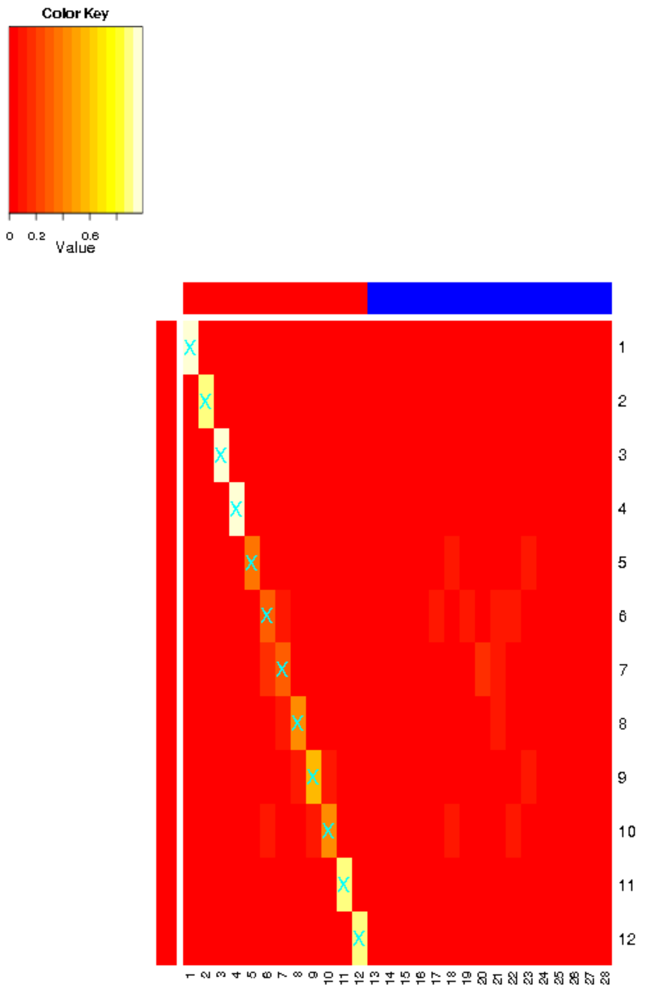


Figure 4.21: Node mapping results: cross testing (A-B) for 3-Layer to 4-Layer Mapping in 4.4.2. Average response matrix after 20 simulations.

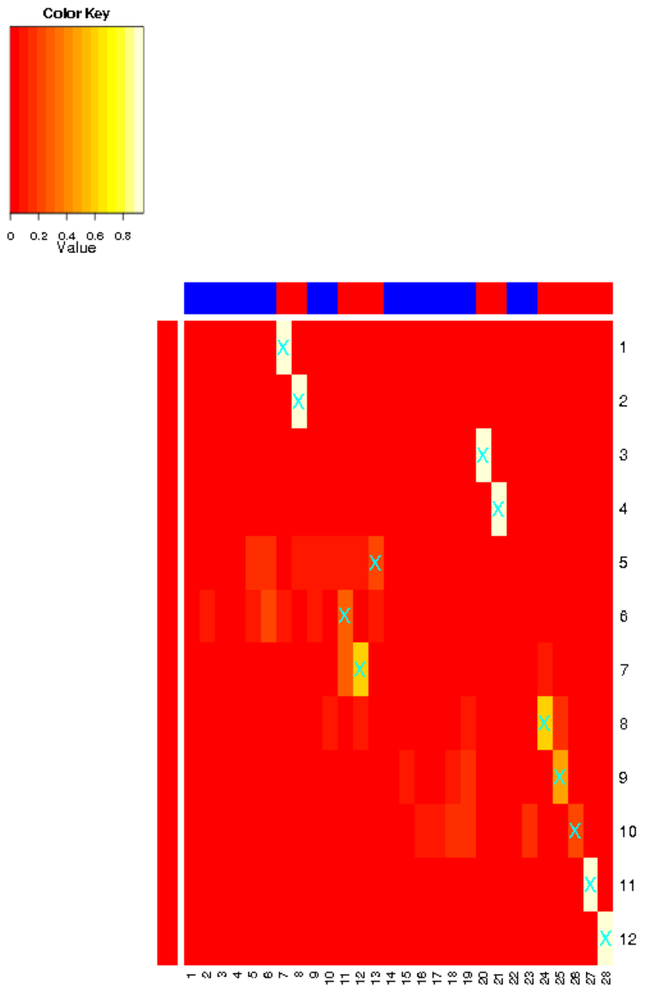


Figure 4.22: Node mapping results: cross testing (A-B) for 3-Layer to 4-Layer Mapping in 4.4.1. Average response matrix after 20 simulations.

4.4.3 Experiment 3: One to Many Relationship Mapping

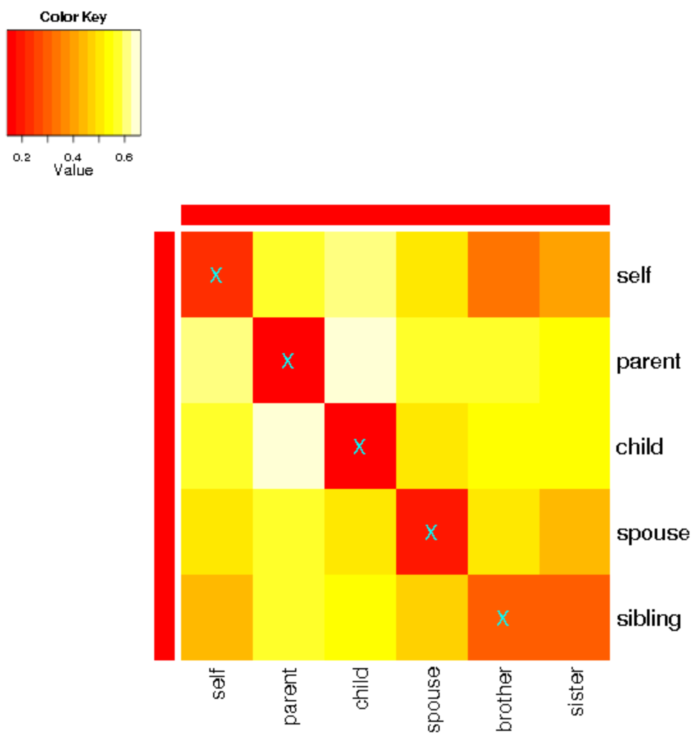
In this experiment, one to many relationship mapping is tested. Although the method used in this dissertation could only derive one to one mapping, one to many mapping could still be explored.

In this experiment, two 3-layer familytrees are to be mapped as shown in Figure 3.5. The English family has no change; however, the “sibling” relationship in the Italian family is replaced by “brother” and “sister”. Specifically, it affects the relationship between “Emilio” and “Lucia”, “Marco” and “Angela”, and “Alfonso” and “Sophia”.

No relationship are explicitly trained in this experiment. Expectation is that “sibling” in the English family is mapped to both “brother” and “sister” in the Italian family. “Christopher” and “Roberto” and “Andrew” and “Pierro” are cross trained.

Average distance matrix at relationship representation layer is shown in the top of Figure 4.23. From the average distance matrix, correct mappings could be derived: “sibling” to “brother” and “sibling” to “sister” have roughly equal distance. Mapping matrix with simulation counts is shown in bottom of Figure 4.23.

Node mapping results are shown in Figure 4.4.3. The top figure shows average response matrix and derived mappings. The derived mappings are all correct when ambiguity is taken into consideration. The bottom figure shows simulation count for each mapping.



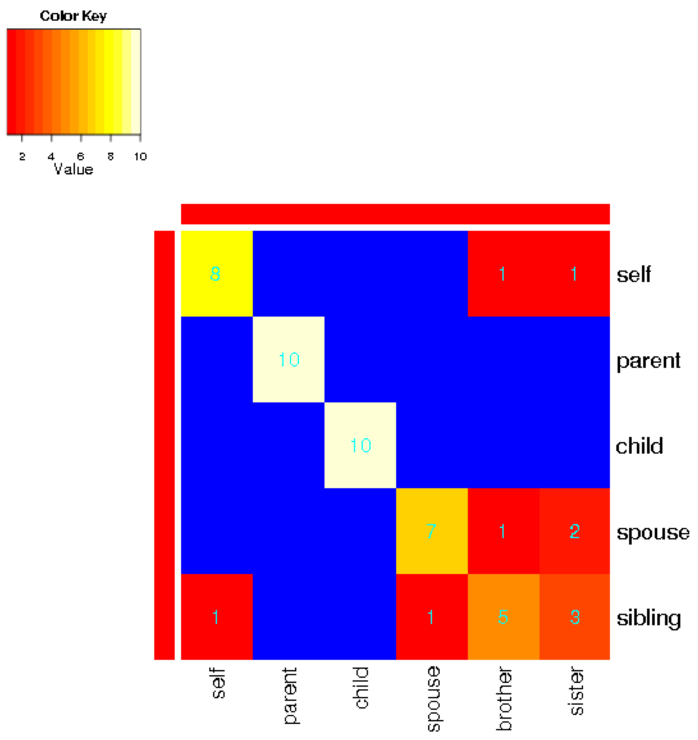


Figure 4.23: Relationship mapping (A-B) for one to many relationship mapping experiment from 4.4.3. Top: Average distance matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.

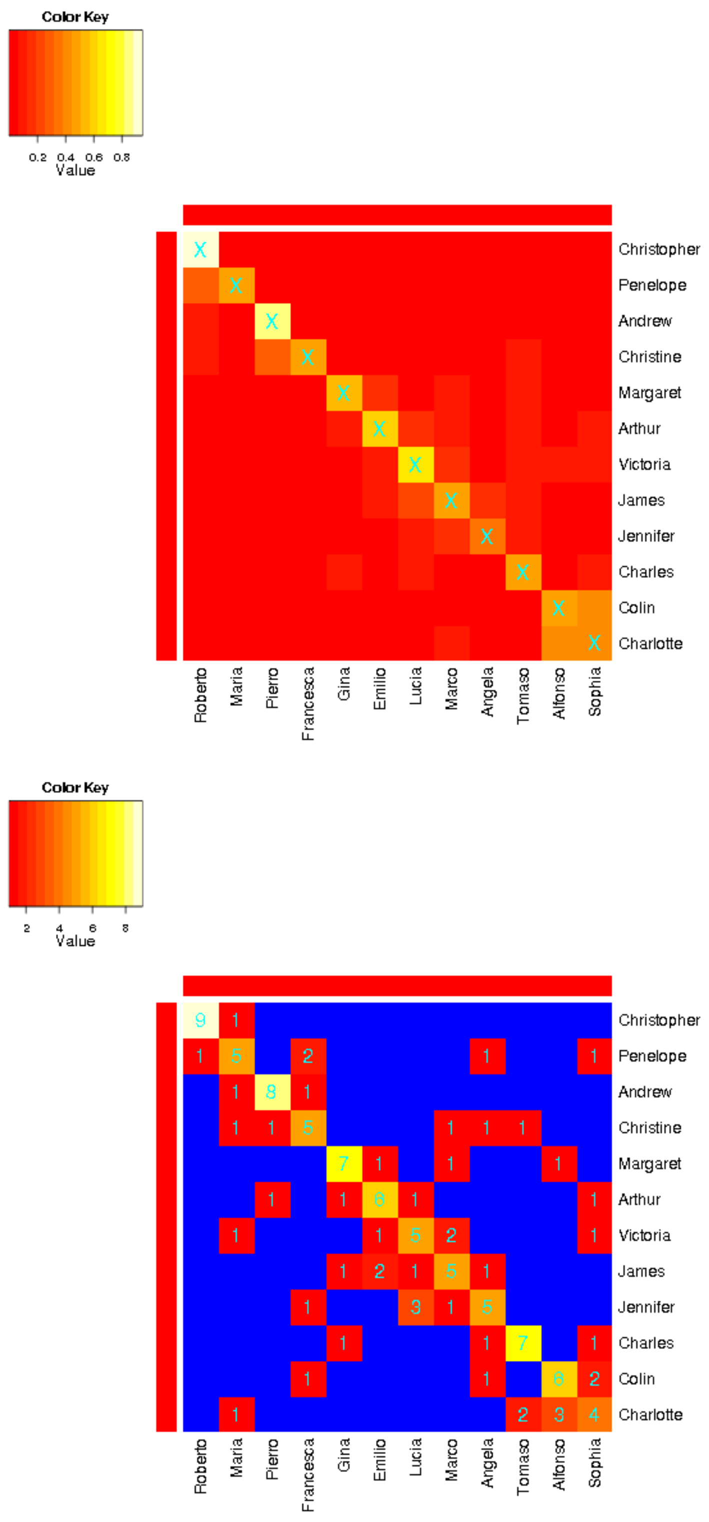


Figure 4.24: Node mapping results: cross testing (A-B) for one to many relationship mapping experiment in 4.4.3. Top: Average response matrix after 20 simulations. Bottom: Mapping matrix with number of simulation in 20 simulations.

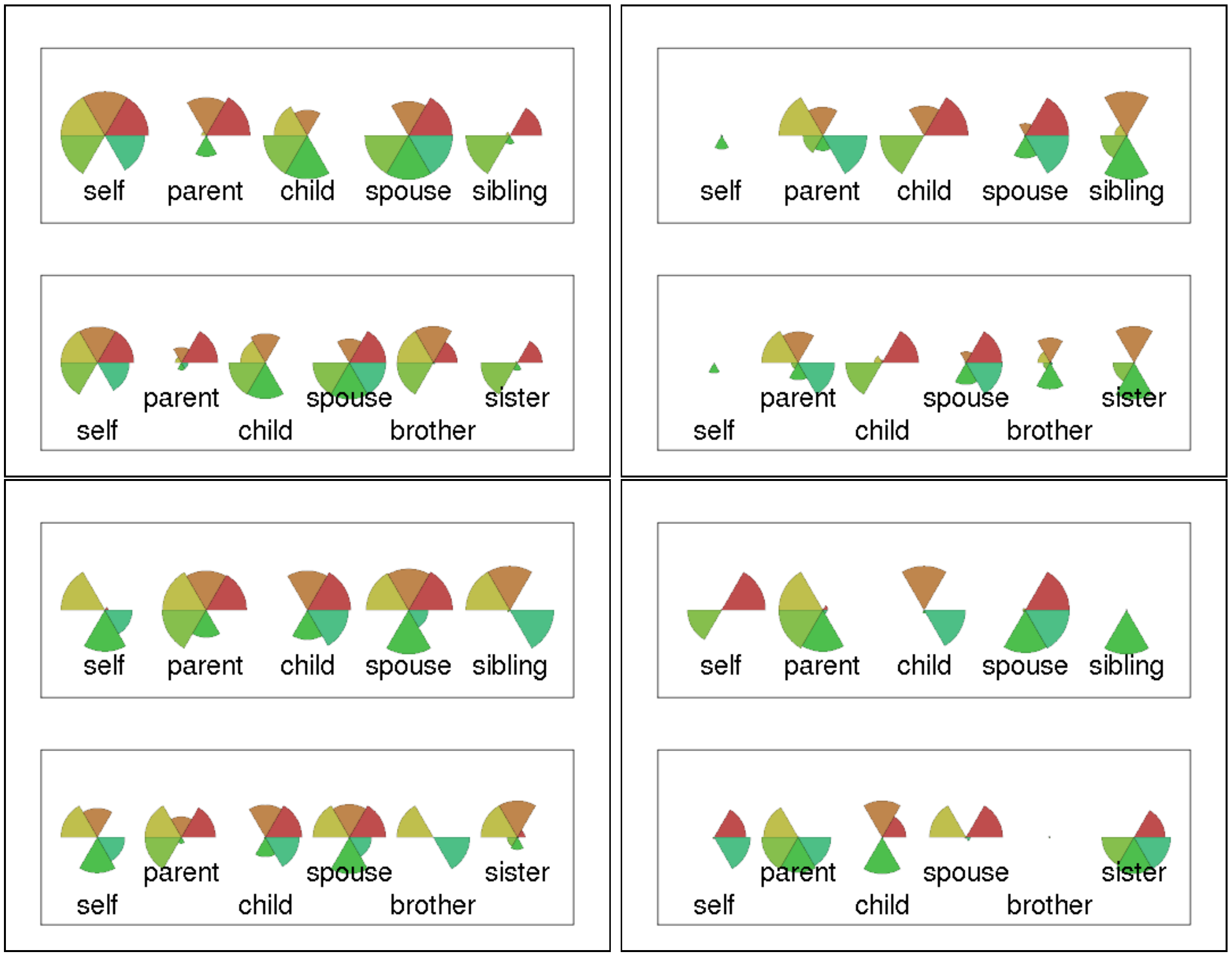


Figure 4.25: Segment plot for output at relationship representation layer for one to many relationship mapping experiment in 4.4.3. The value is shown by the distance from the center to the radius of the segment representing the variable. Each subplot contains result for one simulation. In each subplot, top plot is for one network, bottom plot is for the other. Bottom right: correct mapping could not be found. Others: correct mapping of relationships could be found but not perfect.

4.5 SUMMARY

Experiment results in section 4.1 shows that cross-training helps the network to disambiguate node mappings. Section 4.2 shows that explicit training method works: the training pairs have no ambiguity in testing results, and with more training pairs, the network can get all correct relationship mappings. Section 4.3 shows that the OMNN approach is scalable on 4-layer and 5-layer family tree test cases. One-to-many node mapping and one-to-many relationship mapping are explored in section 4.4, it shows that the network has the ability to get one-to-many mappings. Mapping between family trees with different layers is tested in section 4.4, it shows that the network does disambiguation with cross-training.

With all these test cases, we have enough confidence to apply the network to ontology mapping tasks. Results are shown in next chapter.

5.0 ONTOLOGY MAPPING EVALUATION

Experiments are performed to obtain deeper understanding of the OMNN approach. TSeveral questions to ask are:

- Does the OMNN neural network work in the context of ontology mapping? If it does, how much does it improve from preliminary mappings?
- Does OMNN's performance improve when more training data available?
- How does OMNN perform compare with other systems in OAEI 2009?

Selected OAEI benchmark tests are used to evaluate OMNN approach. All test cases share the same reference ontology, while the test ontologyis different. The reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. In the OMNN approach, classes are treated as items; object properties and data properties are treated as relationships; lastly individuals are not used.

Texture information is used to generate high confident mappings which are then used as cross-training data in OMNN. However OMNN does not focus on how well texture information is used.

In order to compare with other approaches that heavily use texture information, 19 test cases with limited texture information are selected to be used in our experiments. They are test case 249, 257, 258, 265, 259, 266 and their sub-cases.

In all of these test cases, individuals and comments do not exist in test ontology. Only class names and property names are available as textual information in test ontologies. These names are either the same as those in reference ontology or random strings. In the OMNN approach, those pairs having same names are used as training pairs.

5.1 EXPERIMENT 1: TEST CASE 249

In OAEI benchmark test case 249, structure and relationships are the same between the two ontologies. All comments and instances are removed in test ontology. In test ontology, all names are replaced with randomly generated strings. Basically, all of the texture information is removed; only structural information could be used. Four sub-cases are generated with different percentage of "names" replaced with random strings. Test cases 249-2, 249-4, 249-6, 249-8 with 20%, 40% , 60%, and 80% "names" replaced with random strings respectively.

Reference ontology and test ontology in test case 249 are shown in Figure 5.1. Several generic classes have same name in the two ontologies: owl#Thing, foaf#Organization, rdf#List, foaf#Person, owl#Thing, XMLSchema#string, XMLSchema#nonNegativeInteger, XMLSchema#gDay, XMLSchema#gMonth, XMLSchema#gYear, XMLSchema#language. Two properties have same name in the two ontologies: rdf#first,rdf#rest.

Average distance matrix at relationship representation layer is shown in Figure 5.2.

Node mapping results are shown in Figure 5.3. The figure shows average response matrix and derived mappings.

For test case 249-2, 26 classes in test ontology have the same names with their mapping classes in reference ontology, and 51 properties in test ontology have the same names with their mapping properties in reference ontology. These classes and properties are used as cross training pairs in OMNN. Average distance matrix at relationship representation layer is shown in Figure 5.4. Node mapping results are shown in Figure 5.5. The figure shows average response matrix and derived mappings. With more cross training pairs, performance is improved.

The performance of OMNN on all 5 test cases are compared with other systems from 2009 OAEI in Table 5.1. OMNN's performance is better than 8 of the 12 systems, and comparable to the other 4: "aflood", "ASMOV", "Lily", and "RiMOM". The significance test is at the final part of this chapter.

Table 5.1: Experiment Results for Benchmark 249

Systems	249 (no training)			249-2 (80% training)			249-4 (60% training)			249-6 (40% training)			249-8 (20% training)		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
OMNN	0.63	0.63	0.63	0.97	0.97	0.97	0.92	0.92	0.92	0.84	0.84	0.84	0.8	0.8	0.8
aflood	1	0.59	0.74	1	0.97	0.98	1	0.92	0.96	1	0.88	0.94	1	0.84	0.91
AgrMaker	1	0.01	0.02	1	0.79	0.88	1	0.6	0.75	1	0.41	0.58	1	0.22	0.36
aroma	1	0.01	0.02	0.94	0.78	0.85	0.92	0.63	0.75	0.82	0.42	0.56	0.89	0.25	0.39
ASMOV	0.82	0.62	0.71	0.99	0.95	0.97	0.93	0.89	0.91	0.91	0.82	0.86	0.84	0.72	0.78
DSSim	1	0.01	0.02	0.99	0.79	0.88	1	0.6	0.75	0.98	0.41	0.58	0.95	0.22	0.36
GeRoMe	0.25	0.01	0.02	0.96	0.89	0.92	0.83	0.76	0.79	0.73	0.65	0.69	0.68	0.4	0.5
kosimap	1	0.03	0.06	0.95	0.79	0.86	0.9	0.63	0.74	0.83	0.49	0.62	0.75	0.25	0.38
Lily	0.76	0.73	0.74	1	0.97	0.98	0.98	0.91	0.94	0.98	0.87	0.92	0.95	0.82	0.88
MapPSO	0	0	0	0	0	0	0.23	0.23	0.23	0	0	0	0.1	0.1	0.1
RiMOM	0.87	0.61	0.72	1	0.96	0.98	0.98	0.88	0.93	0.93	0.78	0.85	0.81	0.65	0.72
SOBOM	0	0	0	1	0.48	0.65	1	0.35	0.52	1	0.2	0.33	1	0.09	0.17
TaxoMap	0	0	0	0.9	0.29	0.44	0.78	0.22	0.34	0.67	0.16	0.26	0.8	0.08	0.15

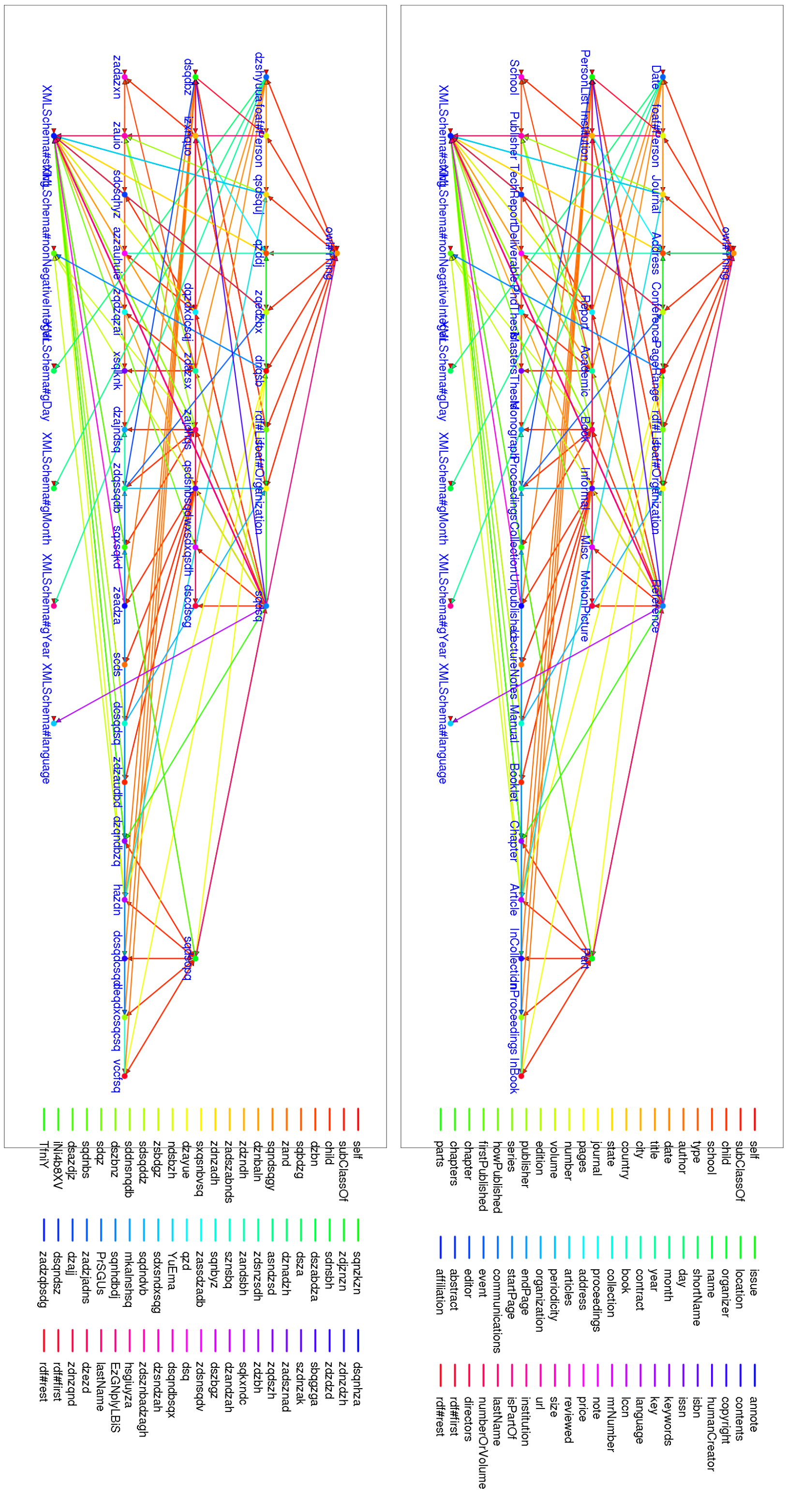


Figure 5.1: Benchmark Test 249. Top: Reference ontology. Bottom: Test ontology.

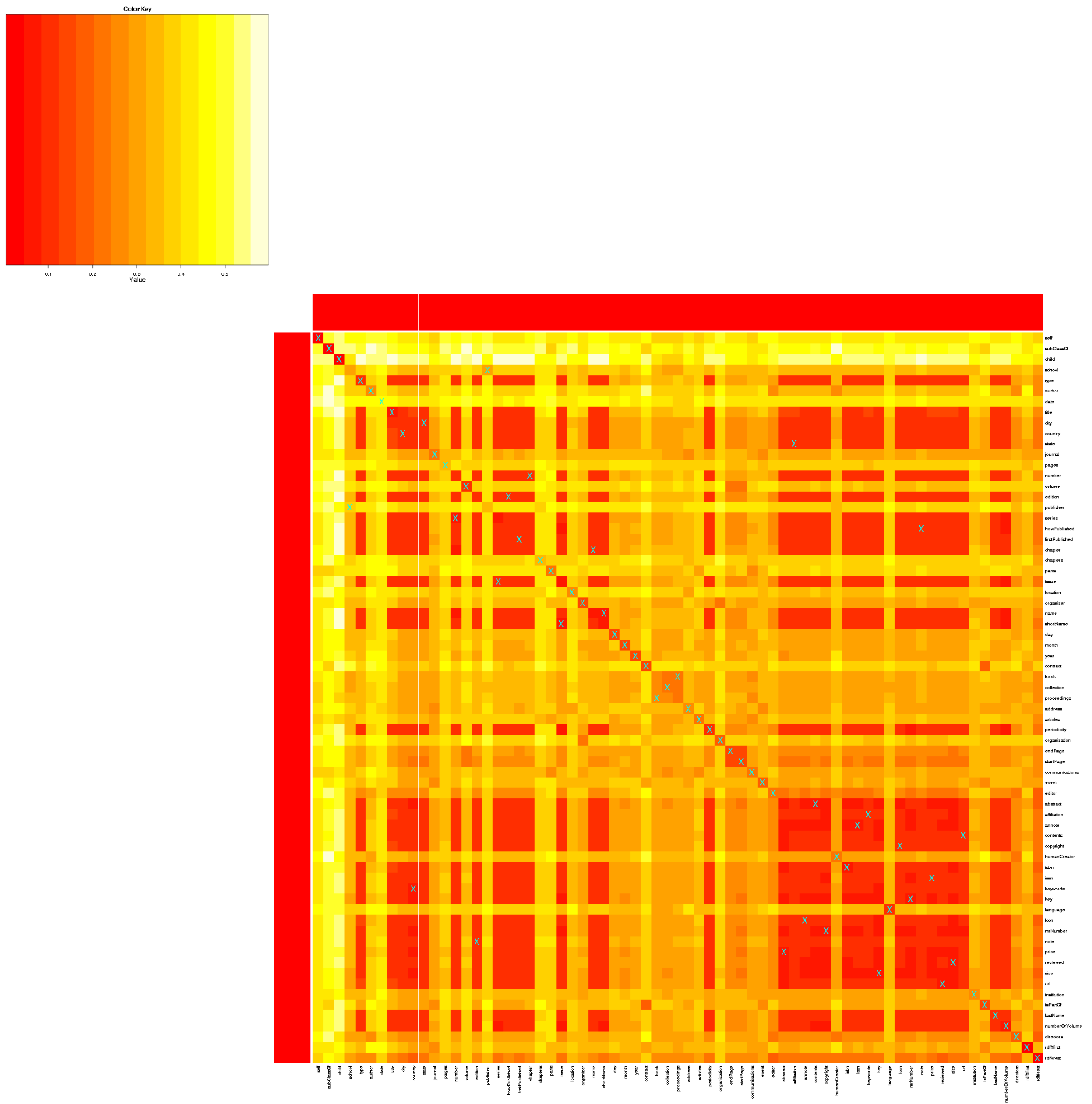


Figure 5.2: Relationship mapping (A-B) for test case 249 in 5.1. Average distance matrix after 20 simulations.

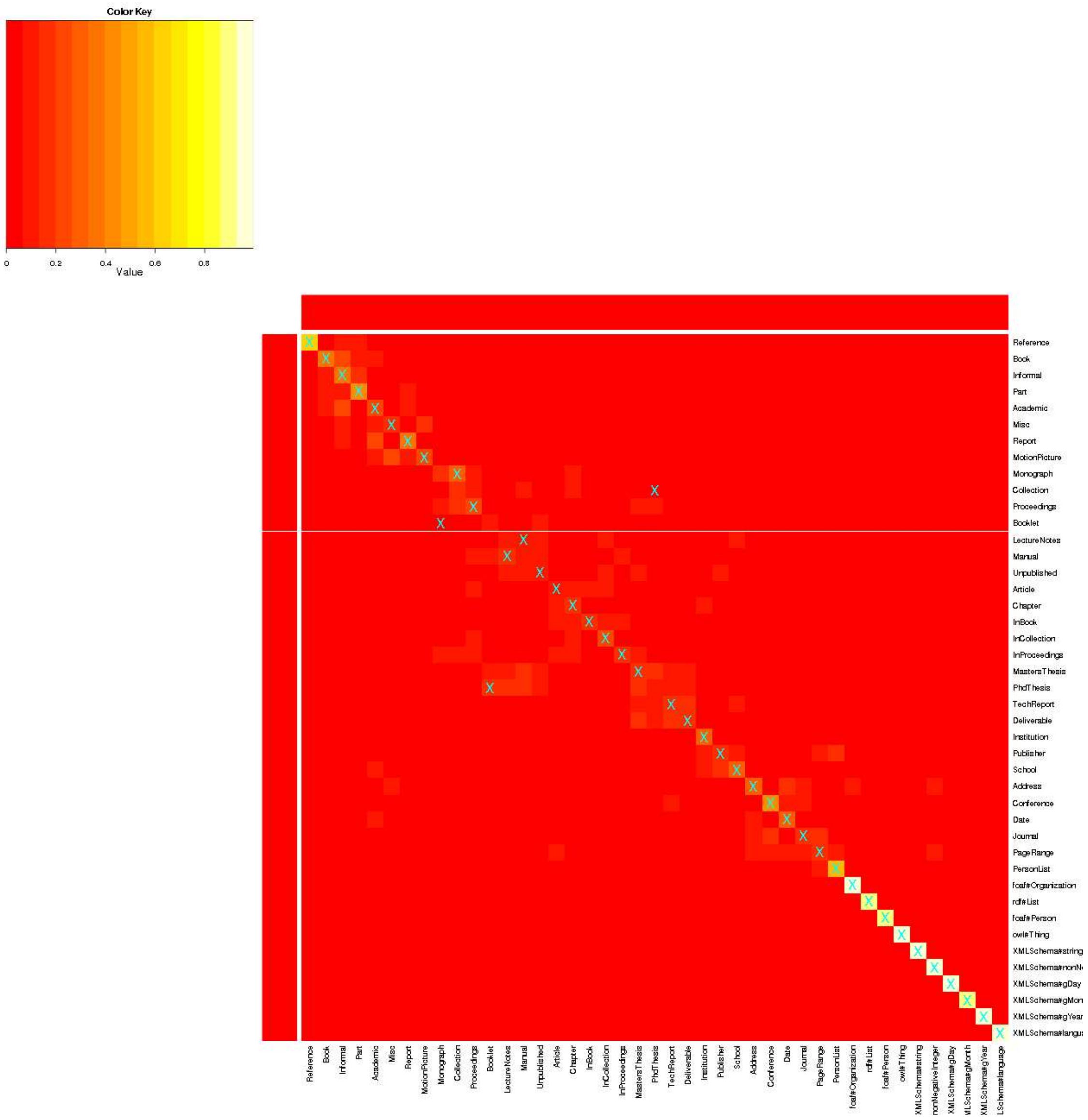


Figure 5.3: Node mapping resutls: cross testing (A-B) for test case 249 in 5.1. Average response matrix after 20 simulations.

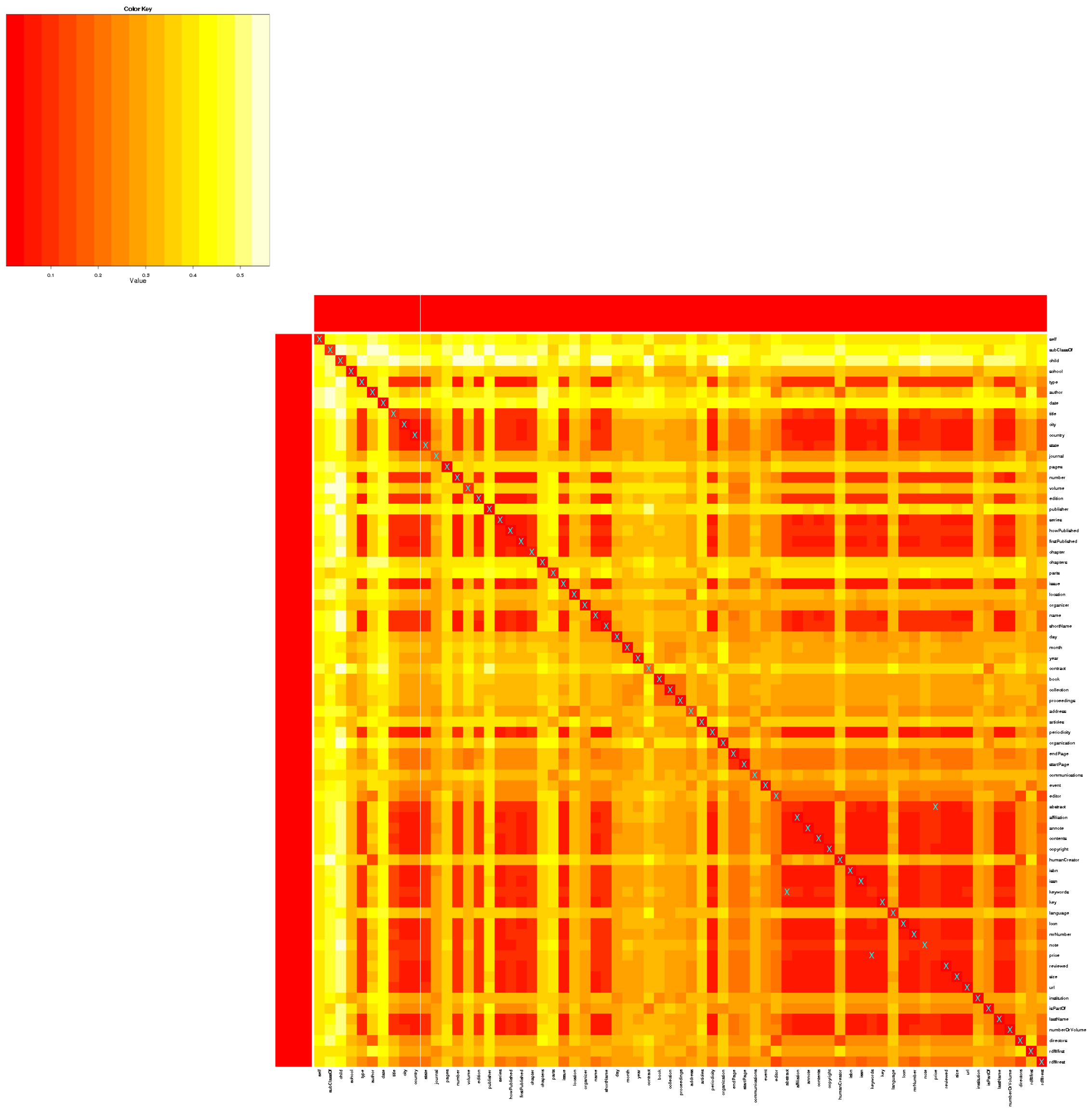


Figure 5.4: Relationship mapping (A-B) for test case 249-2 in 5.1. Average distance matrix after 20 simulations.

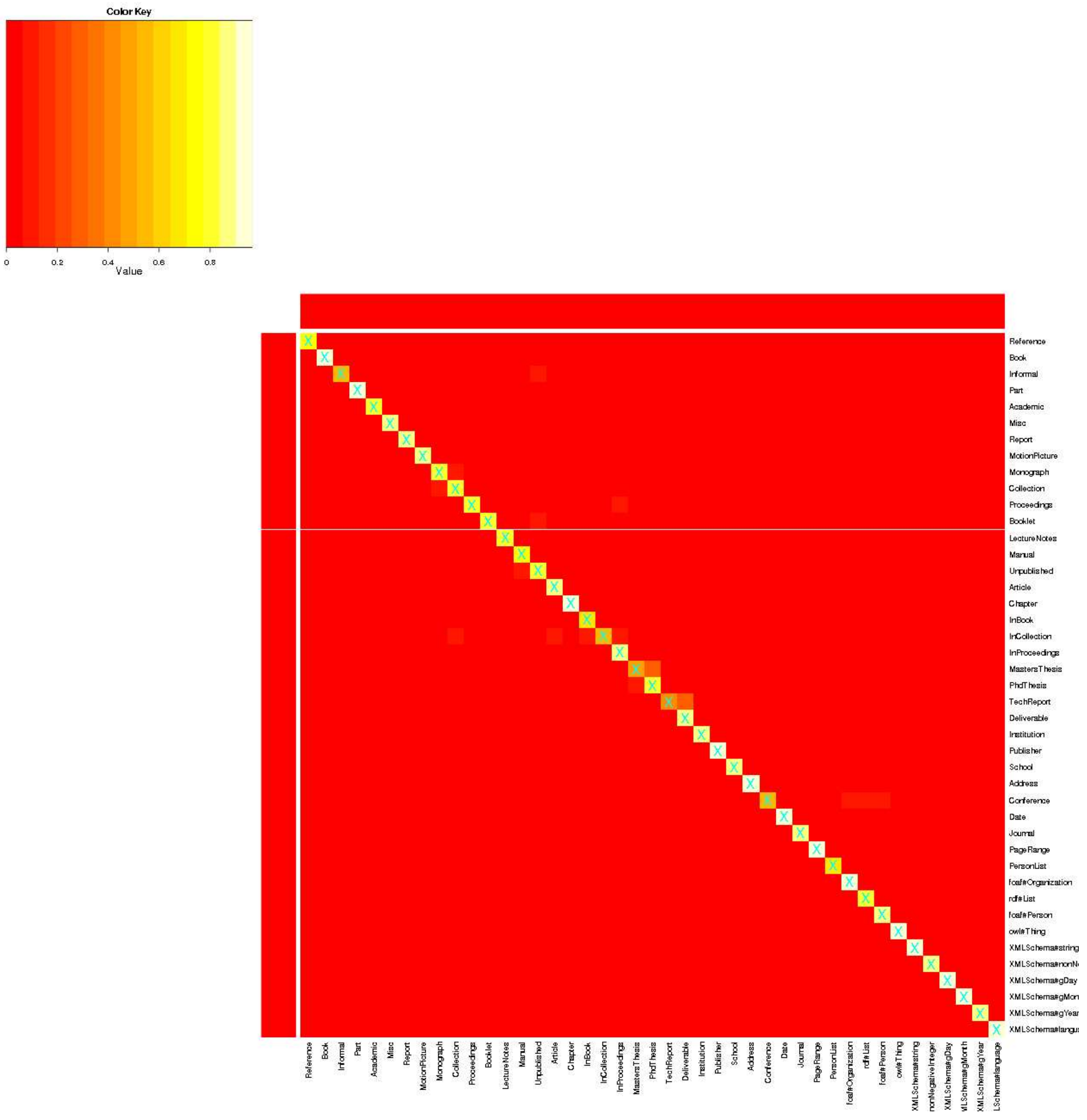


Figure 5.5: Node mapping resutls: cross testing (A-B) for test case 249-2 in 5.1. Average response matrix after 20 simulations.

5.2 EXPERIMENT 2: TEST CASE 257

In test case 257, structure is the same between the two ontologies. All comments, instances are removed from test ontology. Moreover, all relationships (except subClassOf) are removed in test ontology as well, which is the only difference between test case 257 and test case 249.

In test ontology, all names are replaced with randomly generated strings. Basically, all of the texture information is removed, only simple structural information could be used. Four sub-cases are generated with different percentage of ”names“ replaced with random strings. Test cases 257-2, 257-4, 257-6, 257-8 with 20%, 40% , 60%, and 80% ”names“ replaced with random strings respectively.

Reference ontology and test ontology in test case 257 are shown in Figure 5.6. Several generic classes have same name in the two ontologies: owl#Thing, foaf#Organization, rdf#List, owl#Thing.

Node mapping results are shown in Figure 5.7. The figure shows average response matrix and derived mappings.

For test case 257-2, 26 classes in test ontology have the same names with their mapping classes in reference ontology, and 51 properties in test ontology have the same names with their mapping properties in reference ontology. These classes and properties are used as cross training pairs in OMNN. Node mapping results are shown in Figure 5.8. The figure shows average response matrix and derived mappings. With more cross training pairs, performance is improved. The performance of OMNN on all 5 test cases are compared with other systems from 2009 OAEI in Table 5.2. OMNN's performance is better than 7 of the 12 systems, comparable to the other 5: “aflood”, “ASMOV”, “Lily”, “MapPSO” and “RiMOM”. The significance test is given at the final part of this chapter.

Table 5.2: Experiment Results for Benchmark 257

Systems	257 (no training)			257-2 (80% training)			257-4 (60% training)			257-6 (40% training)			257-8 (20% training)		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
OMNN	0.27	0.27	0.27	1	1	1	0.94	0.94	0.94	0.79	0.79	0.79	0.67	0.67	0.67
aflood	1	0.85	0.92	1	0.97	0.98	1	1	1	1	1	1	0.91	0.91	0.91
AgrMaker	0	0	0	1	0.79	0.88	1	0.61	0.76	1	0.42	0.59	1	0.21	0.35
aroma	0	0	0	0.93	0.76	0.84	0.96	0.7	0.81	0.89	0.48	0.62	0.91	0.3	0.45
ASMOV	0.5	0.06	0.11	1	0.97	0.98	1	0.88	0.94	0.88	0.7	0.78	0.83	0.45	0.58
DSSim	0	0	0	0.96	0.79	0.87	1	0.61	0.76	0.93	0.42	0.58	0.88	0.21	0.34
GeRoMe	0	0	0	0.97	0.88	0.92	0.96	0.79	0.87	0.81	0.64	0.72	1	0.36	0.53
kosimap	1	0.06	0.11	0.96	0.79	0.87	0.95	0.61	0.74	0.83	0.45	0.58	0.75	0.27	0.4
Lily	1	0.12	0.21	1	0.97	0.98	1	0.94	0.97	0.87	0.82	0.84	0.85	0.67	0.75
MapPSO	0.24	0.24	0.24	0.88	0.88	0.88	0.94	0.94	0.94	0.61	0.61	0.61	0.52	0.52	0.52
RiMOM	0.58	0.58	0.58	0.85	0.85	0.85	0.88	0.88	0.88	0.73	0.73	0.73	0.55	0.55	0.55
SOBOM	0	0	0	1	0.79	0.88	1	0.61	0.76	1	0.42	0.59	1	0.21	0.35
TaxoMap	0	0	0	0.9	0.85	0.87	0.81	0.67	0.73	0.67	0.48	0.56	0.8	0.24	0.37

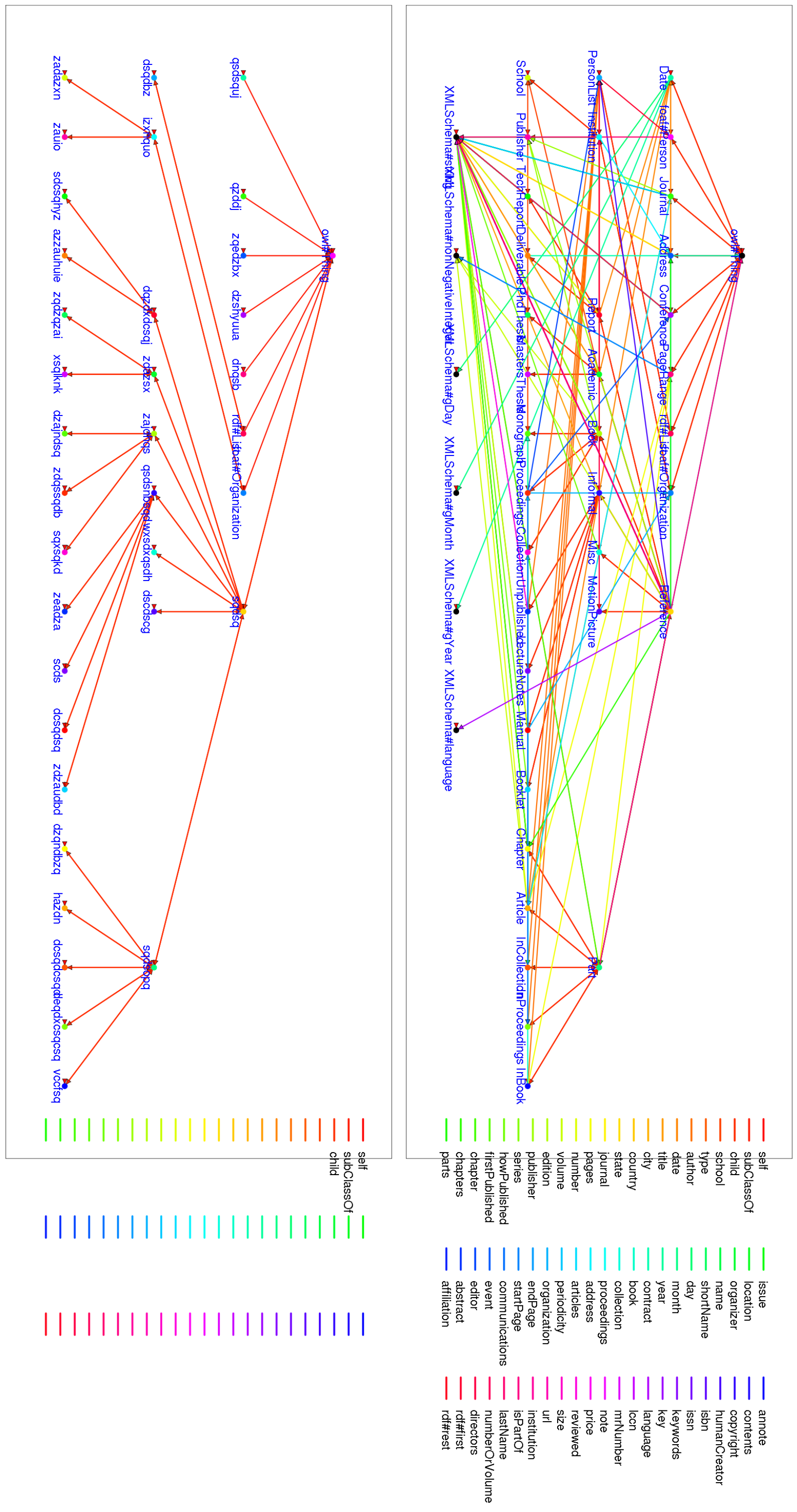


Figure 5.6: Benchmark Test 257. Top: Reference ontology. Bottom: Test ontology.

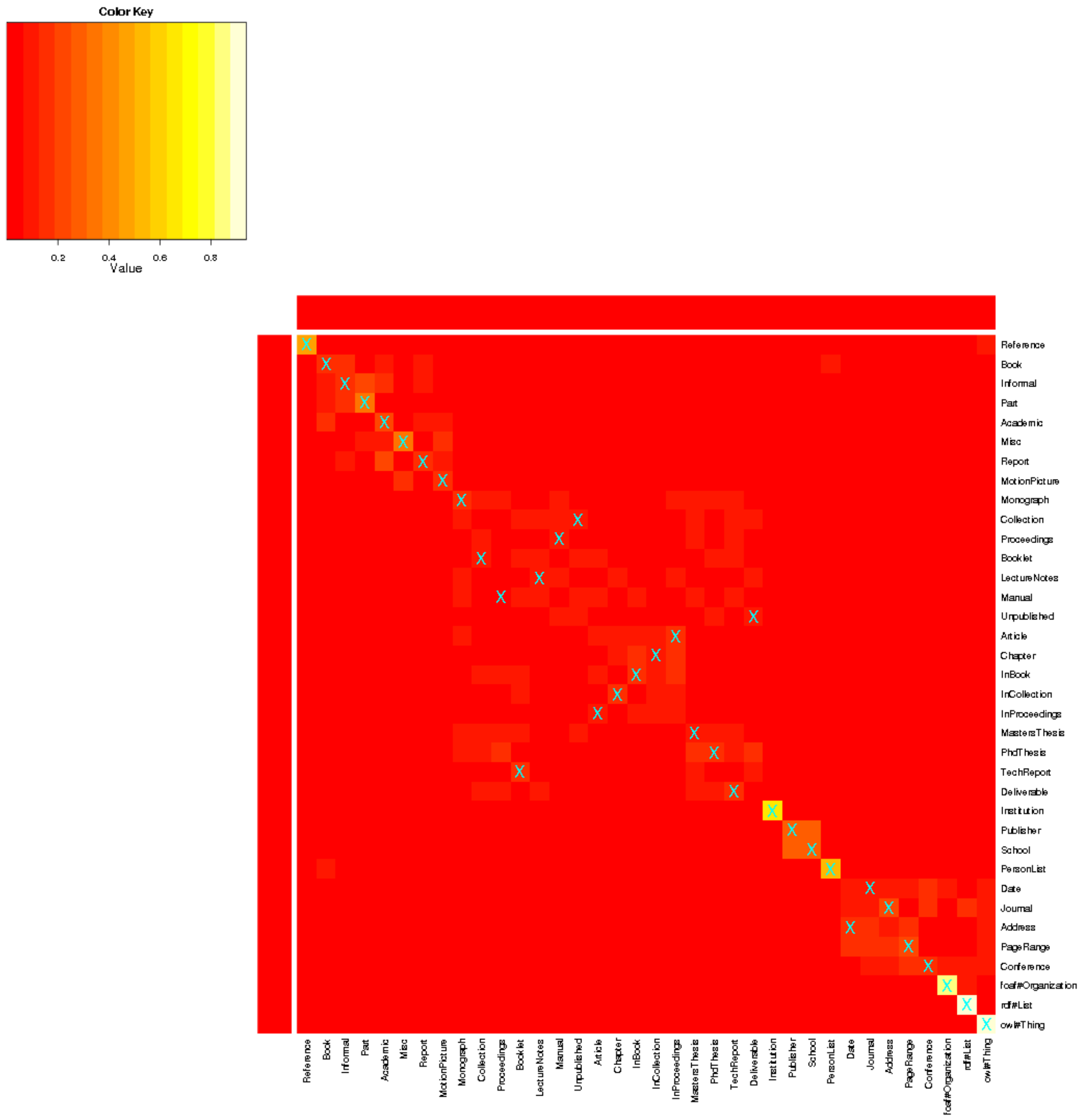


Figure 5.7: Node mapping resutls: cross testing (A-B) for test case 257 in 5.2. Average response matrix after 20 simulations.

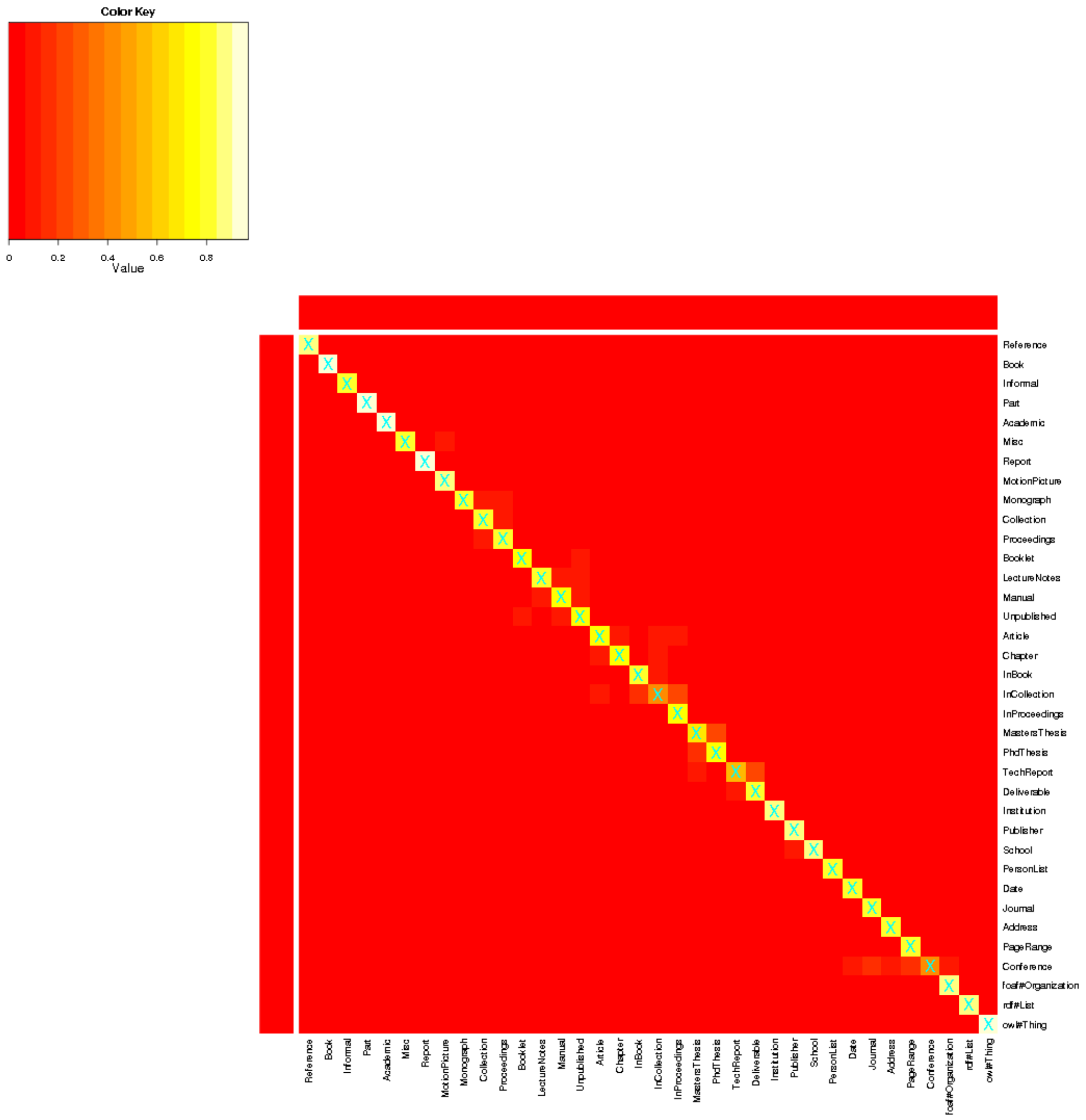


Figure 5.8: Node mapping resutls: cross testing (A-B) for test case 257-2 in 5.2. Average response matrix after 20 simulations.

5.3 EXPERIMENT 3: TEST CASE 258

In test case 258, test ontology has a *flattened* structure based on reference ontology. Compared with reference ontology, four classes are removed: “Academic”, “Book”, “Informal” and “Part”. These four classes have the same parent “Reference”. Only one sibling of them, i.e., “Report”, still exists in test ontology. All their children in reference ontology are now connected with “Reference” directly.

Relationships are kept in the same way as reference ontology. All comments and instances are removed in test ontology. In test ontology, all names are replaced with randomly generated strings. Basically all of the texture information is removed. Only simple structural information can be used. Four sub-cases are generated with different percentage of ”names“, i.e., test cases 258-2, 258-4, 258-6, 258-8 with 20%, 40% , 60%, and 80% of ”names“ replaced with random strings.

Reference ontology and test ontology in test case 258 are shown in Figure 5.9. Several generic classes have the same name in the two ontologies: owl#Thing, foaf#Organization, rdf#List, foaf#Person, owl#Thing, XMLSchema#string, XMLSchema#nonNegativeInteger, XMLSchema#gDay, XMLSchema#gMonth, XMLSchema#gYear, XMLSchema#language. Two properties have the same name in the two ontologies: rdf#first,rdf#rest.

Average distance matrix at relationship representation layer is shown in Figure 5.10.

Node mapping results are shown in Figure 5.11. The figure shows average response matrix and derived mappings.

For test case 258-2, 23 classes in test ontology have same names with their mapping classes in reference ontology, and 51 properties in test ontology have same names with their mapping properties in reference ontology. These classes and properties are used as cross training pairs in OMNN. Average distance matrix at relationship representation layer is shown in Figure 5.12. Node mapping results are shown in Figure 5.13. The figure shows average response matrix and derived mappings. With more cross training pairs, the performance is improved.

The performance of OMNN on all 5 test cases are compared with other systems from 2009 OAEI in Table 5.3.The performance of OMNN is better than 7 of the 12 systems, and comparable to the other 5: "aflood", "ASMOV", "Lily", "MapPSO" and "RiMOM". The significance test is given at the final part of this chapter.

Table 5.3: Experiment Results for Benchmark 258

Systems	258 (no training)			258-2 (80% training)			258-4 (60% training)			258-6 (40% training)			258-8 (20% training)		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
OMNN	0.47	0.47	0.47	0.94	0.94	0.94	0.88	0.88	0.88	0.81	0.81	0.81	0.69	0.69	0.69
aflood	1	0.09	0.17	1	0.9	0.95	0.97	0.78	0.86	0.97	0.67	0.79	1	0.53	0.69
AgrMaker	1	0.01	0.02	1	0.8	0.89	1	0.6	0.75	1	0.41	0.58	1	0.23	0.37
aroma	1	0.01	0.02	0.92	0.76	0.83	0.92	0.61	0.73	0.8	0.4	0.53	0.8	0.22	0.35
ASMOV	0.82	0.63	0.71	0.99	0.94	0.96	0.93	0.88	0.9	0.92	0.82	0.87	0.83	0.69	0.75
DSSim	1	0.01	0.02	0.99	0.8	0.88	1	0.6	0.75	0.97	0.41	0.58	0.95	0.23	0.37
GeRoMe	0.25	0.01	0.02	0.9	0.87	0.88	0.87	0.74	0.8	0.84	0.57	0.68	0.83	0.38	0.52
kosimap	1	0.01	0.02	0.99	0.74	0.85	0.98	0.57	0.72	0.95	0.4	0.56	0.91	0.23	0.37
Lily	0.76	0.56	0.64	0.99	0.96	0.97	0.96	0.88	0.92	0.95	0.83	0.89	0.94	0.8	0.86
MapPSO	0.1	0.1	0.1	0.28	0.28	0.28	0.17	0.17	0.17	0.07	0.08	0.07	0.12	0.12	0.12
RiMOM	0.5	0.15	0.23	0.99	0.76	0.86	0.97	0.81	0.88	0.8	0.74	0.77	0.79	0.58	0.67
SOBOM	0	0	0	1	0.45	0.62	1	0.32	0.48	1	0.17	0.29	1	0.09	0.17
TaxoMap	0	0	0	0.8	0.26	0.39	0.9	0.2	0.33	0.81	0.14	0.24	0.89	0.09	0.16

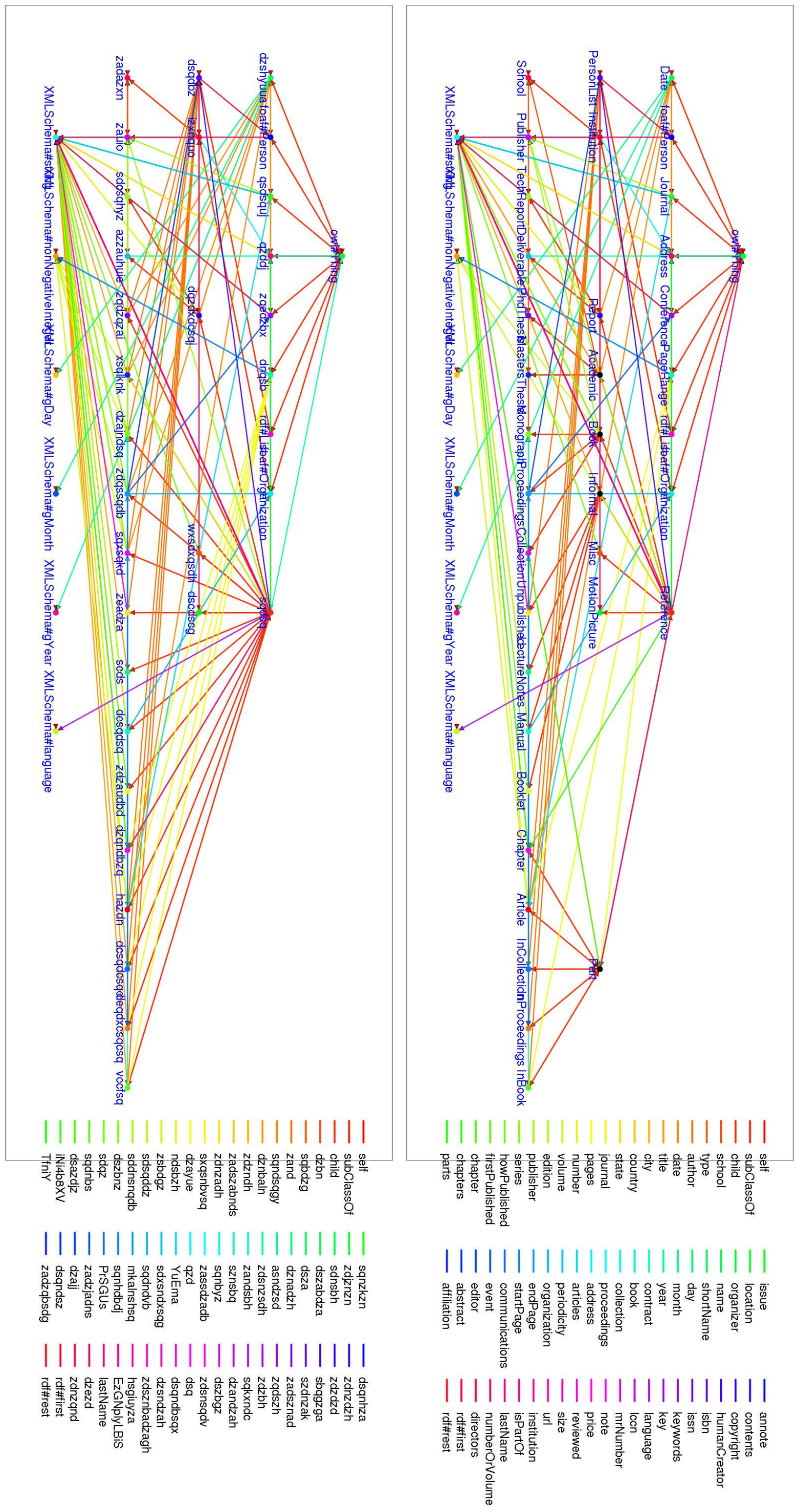


Figure 5.9: Benchmark Test 258. Top: Reference ontology. Bottom: Test ontology.

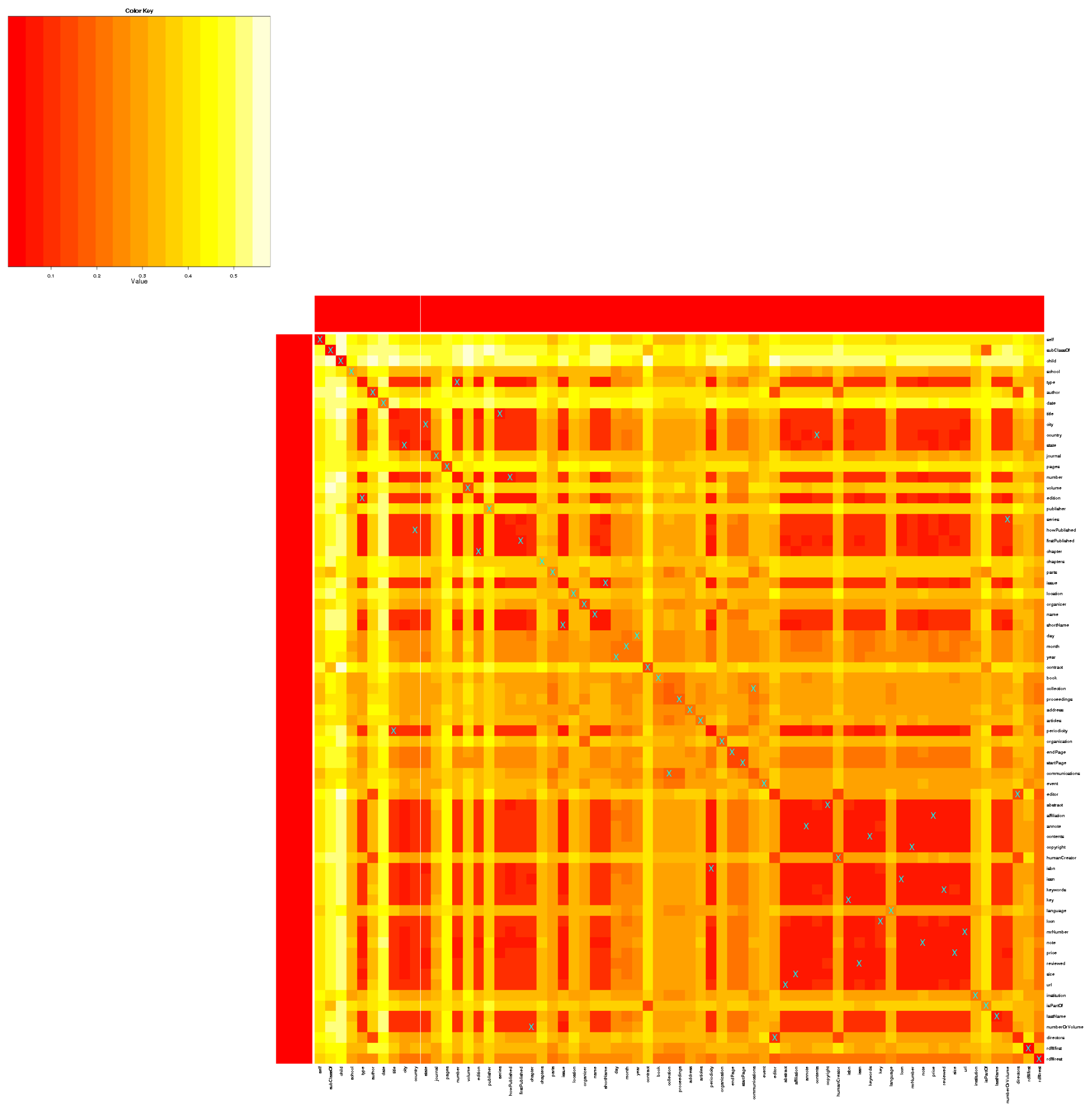


Figure 5.10: Relationship mapping (A-B) for test case 258 in 5.3. Average distance matrix after 20 simulations.

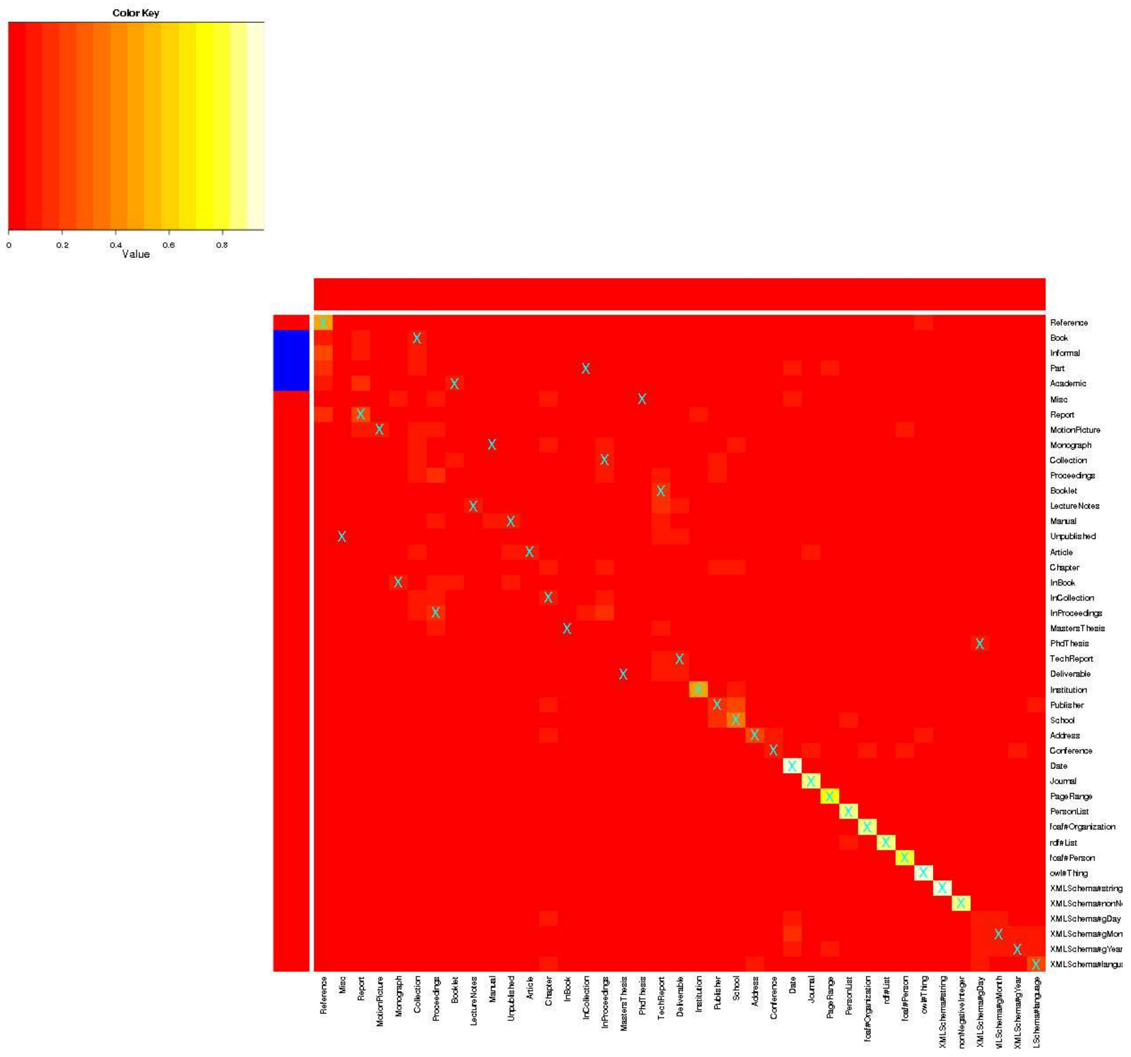


Figure 5.11: Node mapping resutls: cross testing (A-B) for test case 258 in 5.3. Average response matrix after 20 simulations.

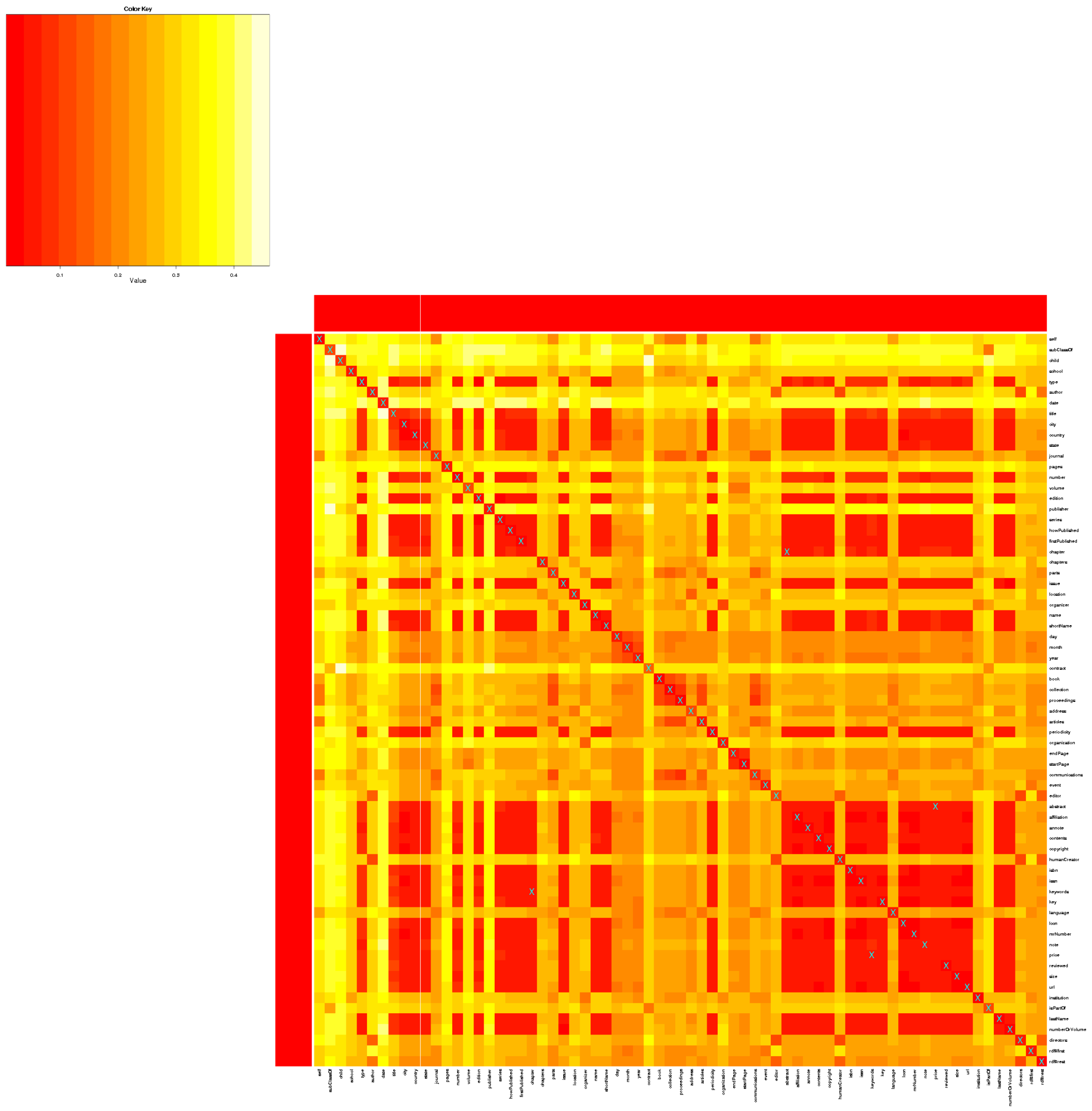


Figure 5.12: Relationship mapping (A-B) for test case 258-2 in 5.3. Average distance matrix after 20 simulations.

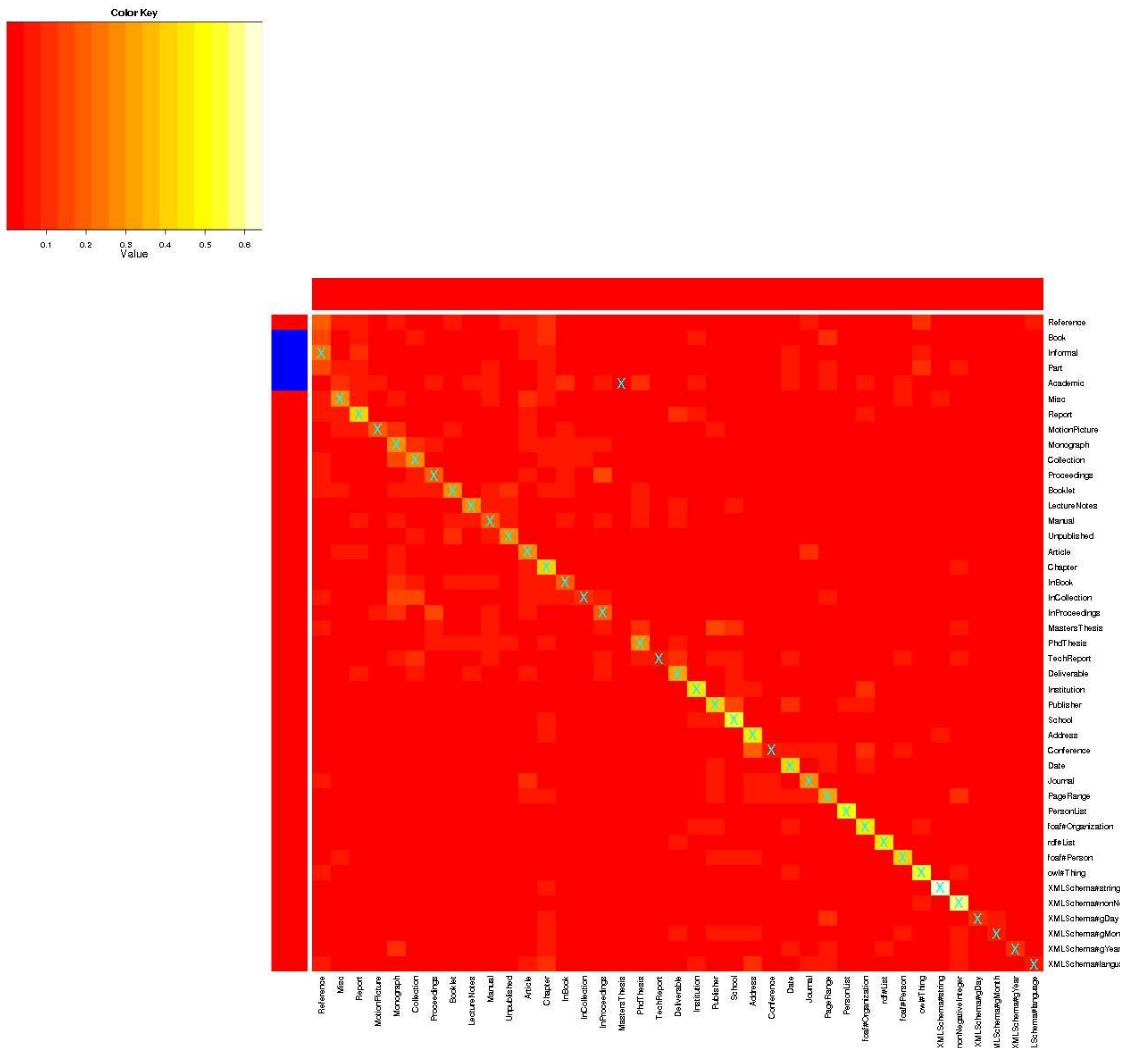


Figure 5.13: Node mapping resutls: cross testing (A-B) for test case 258-2 in 5.3. Average response matrix after 20 simulations.

5.4 EXPERIMENT 4: TEST CASE 259

In test case 259, test ontology has an *expanded* structure based on reference ontology. Hierarchy of the test ontology in test case 259 is shown in Figure 5.14. There are 33 classes added to the existing 33 classes.

Relationships are kept in the same way as reference ontology. All comments, instances are removed in test ontology. In test ontology, all names are replaced with randomly generated strings. Basically all of the texture information is removed, and only simple structural information can be used. One sub-case 259-2 is generated with 20% of ”names“ replaced with random strings. Test cases 259-4, 259-6, and 259-8 are the same as 259-2, which is probably because of human error.

Several generic classes have same names in the two ontologies: owl#Thing, foaf#Organization, rdf#List, foaf#Person, owl#Thing, XMLSchema#string, XMLSchema#nonNegativeInteger, XMLSchema#gDay, XMLSchema#gMonth, XMLSchema#gYear, XMLSchema#language. Two properties have same names in the two ontologies: rdf#first,rdf#rest.

Average distance matrix at relationship representation layer is shown in Figure 5.15. Node mapping results are shown in Figure 5.16. The figure shows average response matrix and derived mappings.

For test case 259-2, 26 classes in test ontology have same names with their mapping classes in reference ontology, and 51 properties in test ontology have same names with their mapping properties in reference ontology. These classes and properties are used as cross training pairs in OMNN. Average distance matrix at relationship representation layer is shown in Figure 5.17. Node mapping results are shown in Figure 5.18. The figure shows average response matrix and derived mappings. With more cross training pairs, the performance of OMNN is improved. The performance of OMNN on the 2 test cases compared with other systems from 2009 OAEI in Table 5.4. The performance of OMNN is better than 9 of the 12 systems and comparable to the other 3: "aflood", "ASMOV" and "Lily". The significance test is given at the final part of this chapter.

Table 5.4: Experiment Results for Benchmark 259

Systems	259 (no training)			259-2 (80% training)		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
OMNN	0.4	0.4	0.4	0.95	0.95	0.95
aflood	0.86	0.06	0.11	0.98	0.92	0.95
AgrMaker	1	0.01	0.02	1	0.79	0.88
aroma	0.86	0.06	0.11	0.9	0.78	0.84
ASMOV	0.81	0.62	0.7	0.97	0.93	0.95
DSSim	0.88	0.07	0.13	0.98	0.8	0.88
GeRoMe	0.28	0.05	0.08	0.91	0.9	0.9
kosimap	1	0.03	0.06	0.92	0.78	0.84
Lily	0.91	0.73	0.81	0.97	0.94	0.95
MapPSO	0.04	0.04	0.04	0.23	0.23	0.23
RiMOM	0.53	0.16	0.25	0.86	0.7	0.77
SOBOM	0	0	0	1	0.44	0.61
TaxoMap	0	0	0	0.87	0.28	0.42

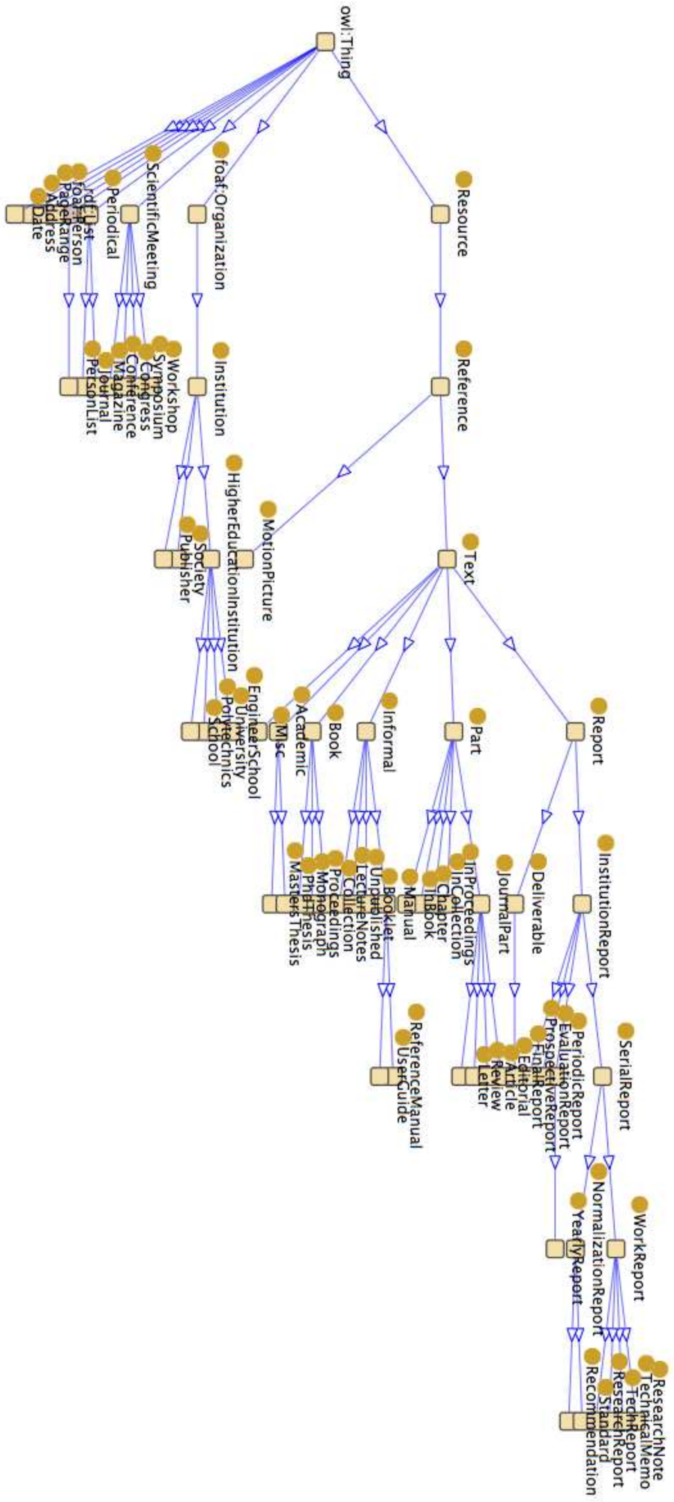


Figure 5.14: Hierarchy of classes in Benchmark Test 259

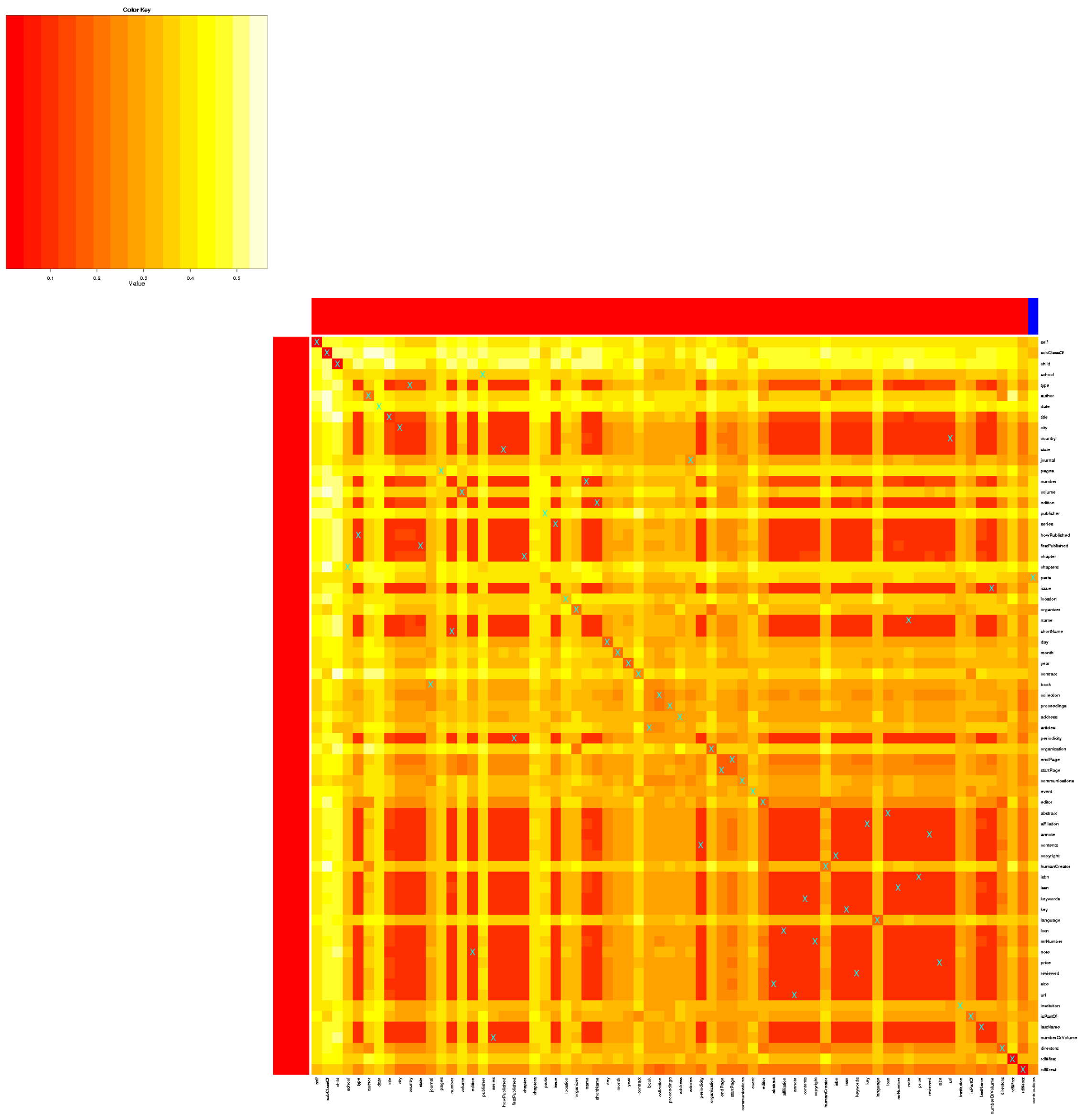


Figure 5.15: Relationship mapping (A-B) for test case 259 in 5.4. Average distance matrix after 20 simulations.

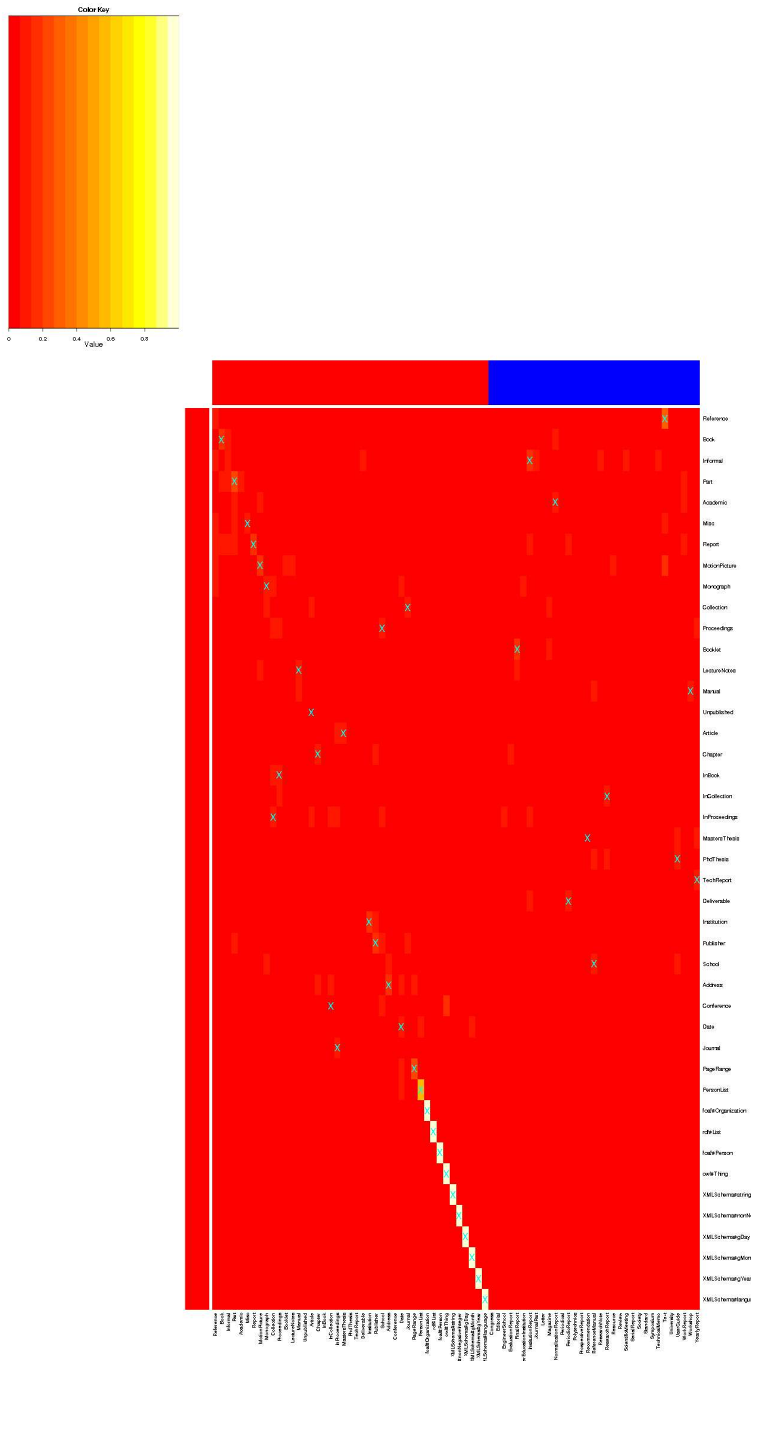


Figure 5.16: Node mapping results: cross testing (A-B) for test case 259 in 5.4. Average response matrix after 20 simulations.

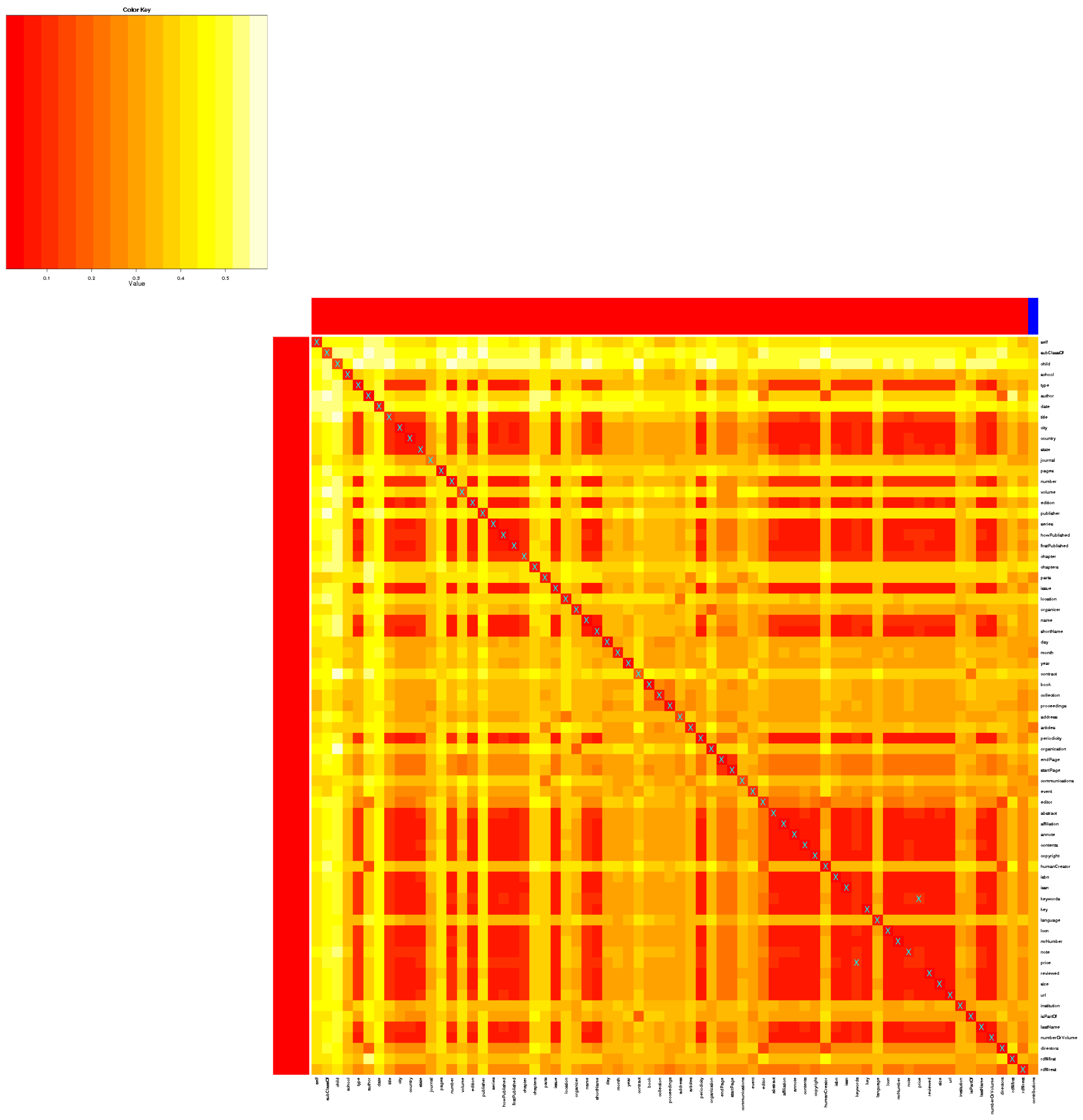


Figure 5.17: Relationship mapping (A-B) for test case 259-2 in 5.4. Average distance matrix after 20 simulations.

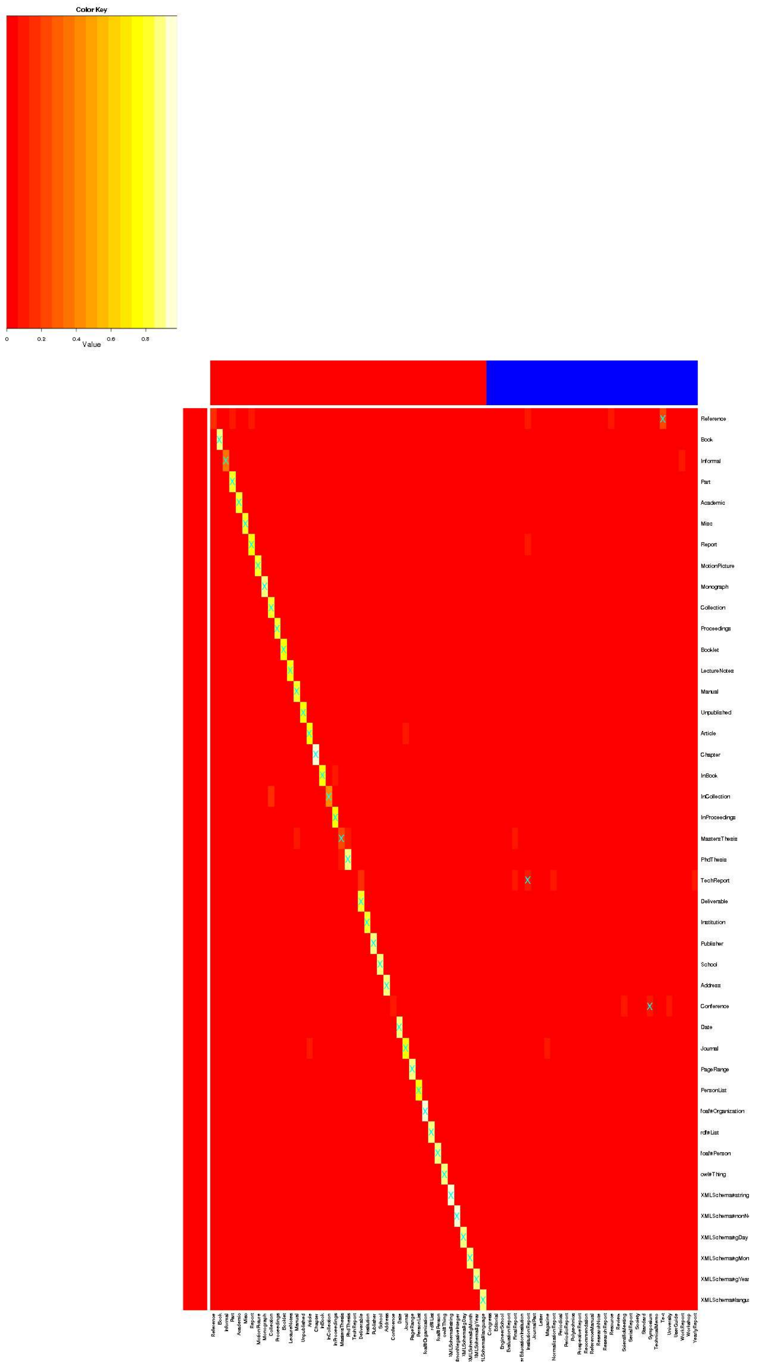


Figure 5.18: Node mapping results: cross testing (A-B) for test case 259-2 in 5.4. Average response matrix after 20 simulations.

5.5 EXPERIMENT 5: TEST CASE 265

Test case 265 is similar with test case 258, where struncture is *flatteren* compared with reference ontology. The difference is, in 265 all relationships (except subClassOf) are removed in test ontology. As other test cases, all comments, instances are removed from test ontology. In test ontology, all names are replaced with randomly generated strings. Basically all of the texture information is removed, only simple structural information could be used.

Node mapping results are shown in Figure 5.19. The figure shows average response matrix and derived mappings.

The performance of OMNN compared with other systems from 2009 OAEI is shown in Table 5.5. The performance of OMNN is better than 8 of the 12 systems, and comparable to the other 4: “aflood”, “ASMOV”, “Lily”, and “RiMOM”. The significance test is given at the final part of this chapter.

Table 5.5: Experiment Results for Benchmark 265

Systems	265 (no training)		
	Precision	Recall	F-Measure
OMNN	0.28	0.28	0.28
aflood	0.8	0.14	0.24
AgrMaker	0	0	0
aroma	0	0	0
ASMOV	0.4	0.07	0.12
DSSim	0	0	0
GeRoMe	0	0	0
kosimap	0.67	0.07	0.13
Lily	0.8	0.14	0.24
MapPSO	0.1	0.1	0.1
RiMOM	0.33	0.1	0.15
SOBOM	0	0	0
TaxoMap	0	0	0

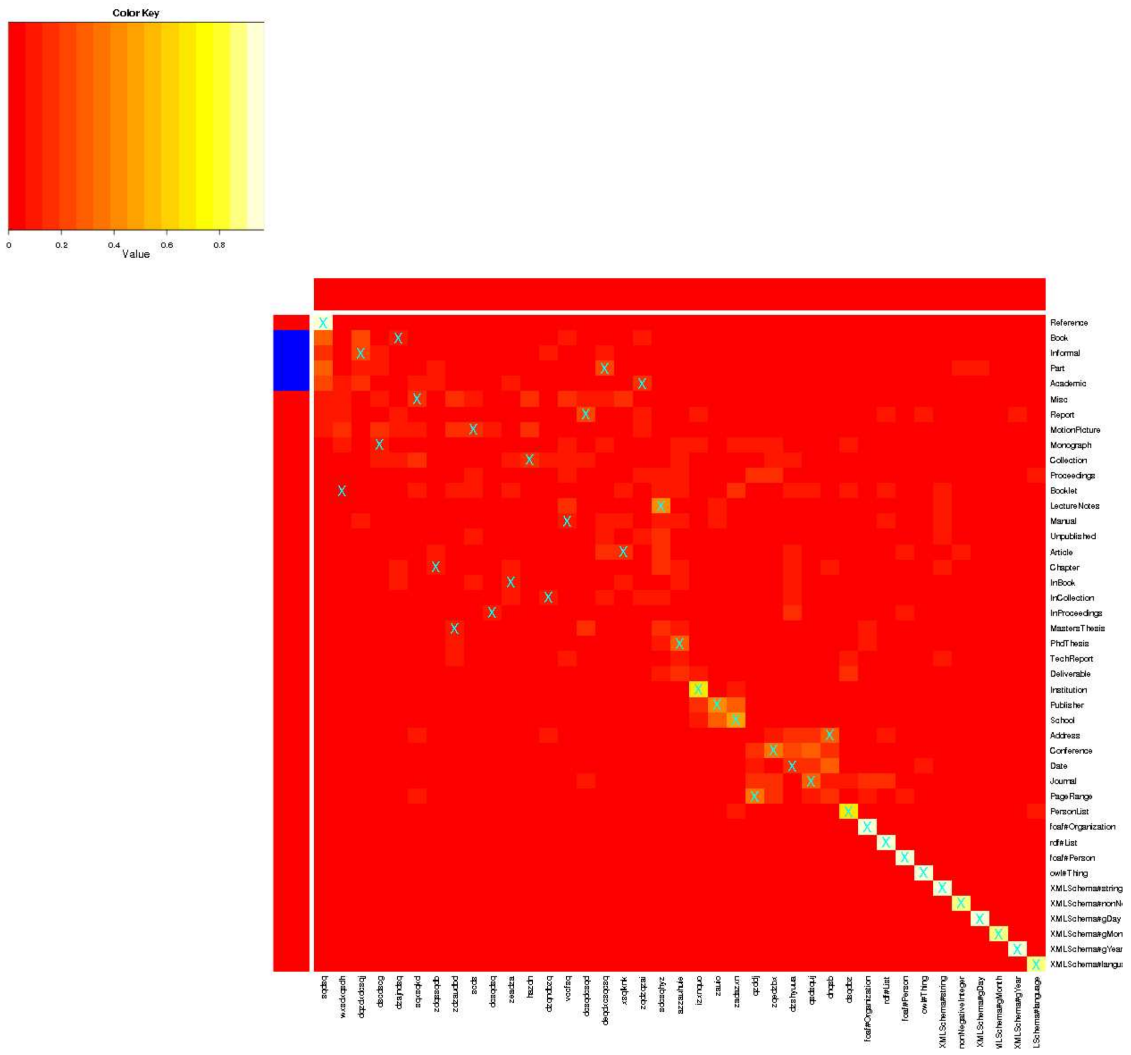


Figure 5.19: Node mapping results: cross testing (A-B) for test case 265 in 5.5. Average response matrix after 20 simulations.

5.6 EXPERIMENT 6: TEST CASE 266

Test case 266 is similar with test case 259, where structure is *expanded* compared with reference ontology. Hierarchy of test ontology in test case 266 is shown in Figure 5.14. There are 33 classes added to the existing 33 classes.

The difference from test case 259 is, all relationships (except subClassOf) are removed in test ontology in test case 266. Like other test cases, all comments and instances are removed from test ontology, and all names are replaced with randomly generated strings. Basically all of the texture information is removed, only simple structural information can be used.

Node mapping results are shown in Figure 5.20. The figure shows average response matrix and derived mappings.

The performance of OMNN compared with other systems from 2009 OAEI is shown in Table 5.6. The performance of OMNN is better than 8 of the 12 systems, and comparable to the other 4: “aflood”, “ASMOV”, “Lily” and “RiMOM”. The significance test is given at the final part of this chapter.

Table 5.6: Experiment Results for Benchmark 266

Systems	266 (no training)		
	Precision	Recall	F-Measure
OMNN	0.21	0.21	0.21
aflood	0.5	0.06	0.11
AgrMaker	0	0	0
aroma	0	0	0
ASMOV	0.4	0.06	0.1
DSSim	0	0	0
GeRoMe	0	0	0
kosimap	0.67	0.06	0.11
Lily	0.5	0.09	0.15
MapPSO	0.06	0.06	0.06
RiMOM	0.18	0.06	0.09
SOBOM	0	0	0
TaxoMap	0	0	0

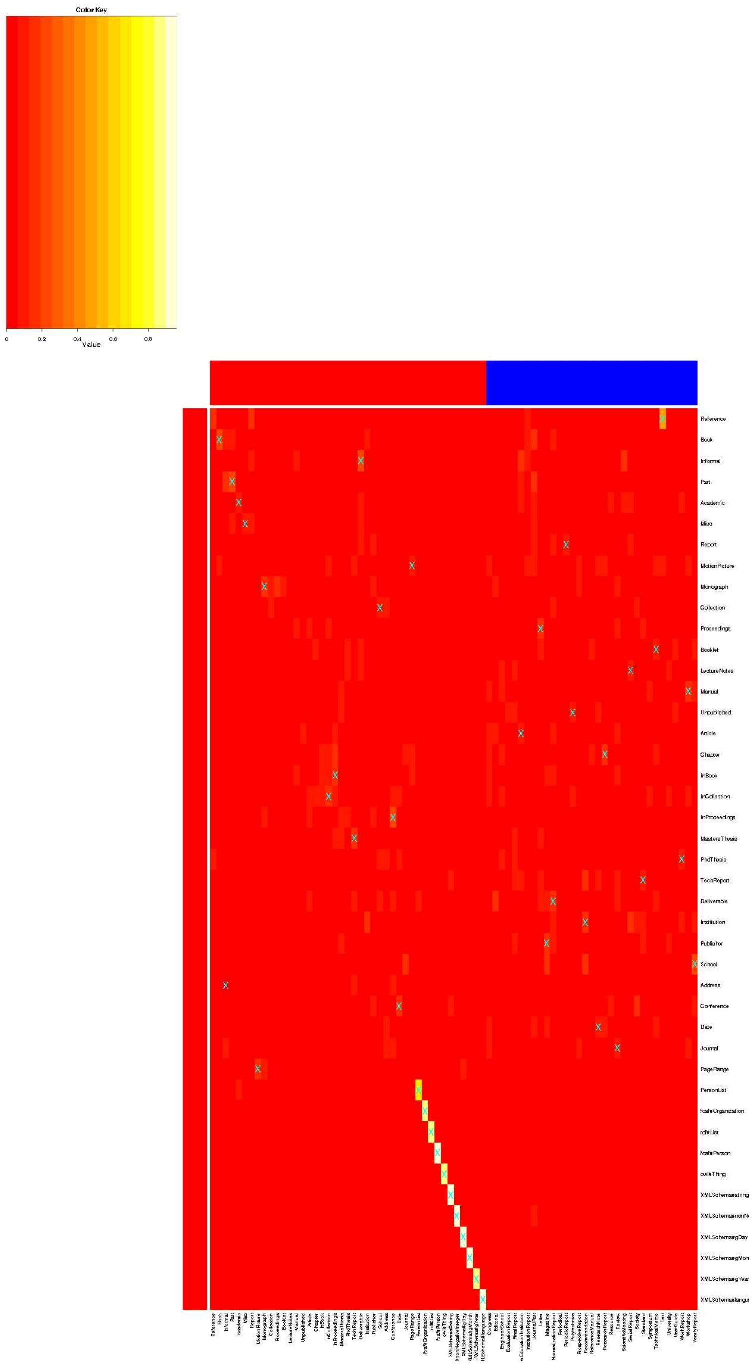


Figure 5.20: Node mapping results: cross testing (A-B) for test case 266 in 5.6. Average response matrix after 20 simulations.

5.7 SIGNIFICANCE TEST

To get a more meaningful comparison, the Wilcox test is performed to compare OMNN with the other 12 systems on precision, recall and f-measure. The result is shown in Table 5.7.

OMNN's recall is significantly better than 10 of the systems, and there is no significant difference with the recall of "aflood" and "Lily". OMNN's F-measure is significantly better than 10 of the systems. There is no significant difference with the F-measure of "aflood". "Lily" has significantly better F-measure than OMNN.

OMNN's precision is significantly better than "GeRoMe", "MapPSO", and "TaxoMap". OMNN's precision has no significant difference with the precision of "aroma", "RiMOM", and "SOBOM". The precision of OMNN is significantly worse than that of "aflood", "AgrMaker", "ASMOM", "DSSim", "kosimap", and "Lily".

Current OMNN approach does not have threshold to remove unlikely mappings based on similarity of distance. Hence its precision is not very strong. For the tests mostly with low recalls, OMNN performs well on recall. Even though OMNN has low precision, its F-measure is very strong.

Similarity scatter plot is shown in Figure 5.22. Precison-recall curve is shown in Figure 5.22. From cutoff threshold 0.95 to 0.9, recall doesn't change much (from 0.29 to 0.37), but precision drops dramatically from 0.99 to 0.76. Similarity scatter plot shows that from 0.9 to 0.95, there are many false positives for cases 249, 258, and their sub-cases.

Increasing threshold neither improves overall performance nor changes significance test results much. Using cutoff threshold 0.1 or 0.15 improves the precision a little. However, the precision difference between "OMNN" and "AgrMaker", "OMNN" and "DSSim" is not significant and it does not change other comparisons. Using cutoff threshold larger than 0.15 hurts recall.

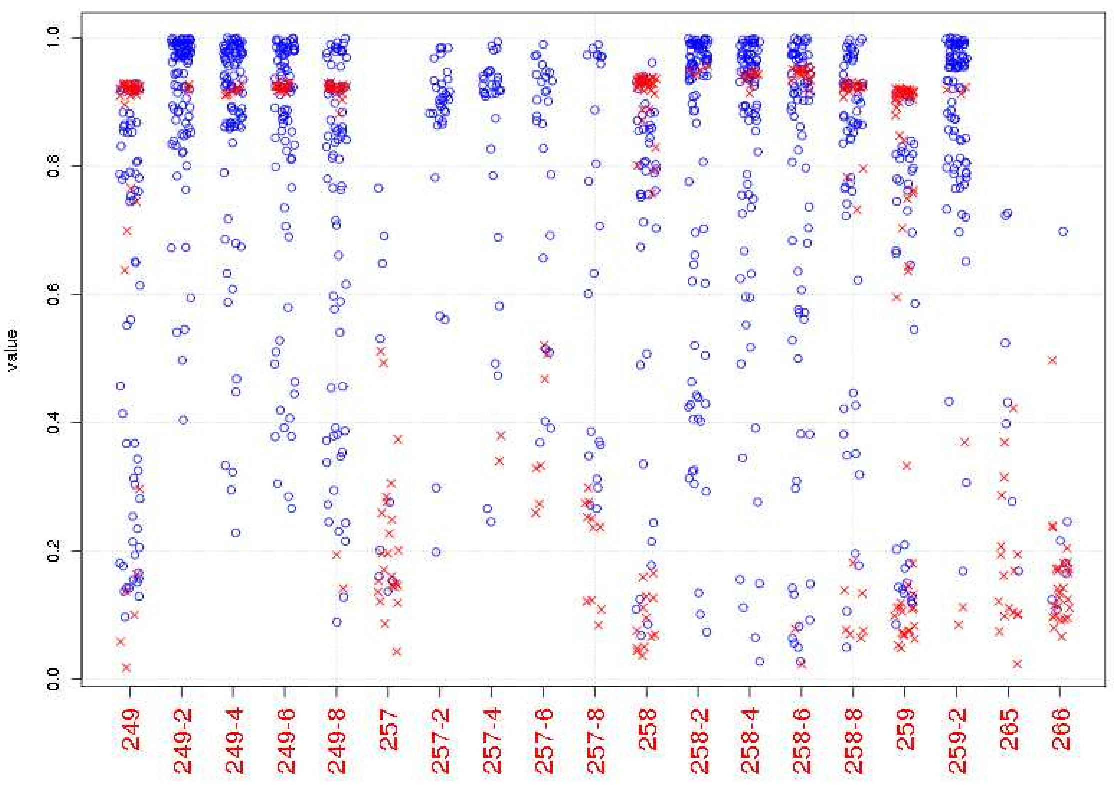


Figure 5.21: Similarity scatterplot for OAEI test cases. Blue circles represent correct mappings; Red crosses represent wrong mappings.

5.8 SUMMARY

All previous ontology mapping result shows that OMNN works in the contexty of ontology mapping. In response to the question: "How much does OMNN improve from preliminary mappings", recall comparison between OMNN and preliminary mapping result is listed in

Table 5.8. Since there is no threshold in OMNN approach, precision, recall and F-measure have the same value for each test case. It's also true for preliminary mapping, so only one metric needs to be compared. From Table 5.8, OMNN always has better performance than preliminary mapping. The average recall improvement on the 19 test cases is 0.33. Wilcox test shows that OMNN is significantly better than preliminary mapping.

To answer the question "Does OMNN's performance improve when more training data available", we can see OMNN gets better performance with more training data from part of the data from Table 5.8 which is plotted as Figure 5.23,

Results from section 5.7 answers the question "How does OMNN perform compared with other systems." OMNN approach is competitive to top ranking systems from OAEI 2009. The evaluation is performed on 19 OAEI 2009 benchmark test cases. Significance test shows OMNN's F-measure is significantly better than 10 of the systems, with no significant difference with the F-measure of "aflood", and "Lily" has significantly better F-measure than OMNN.

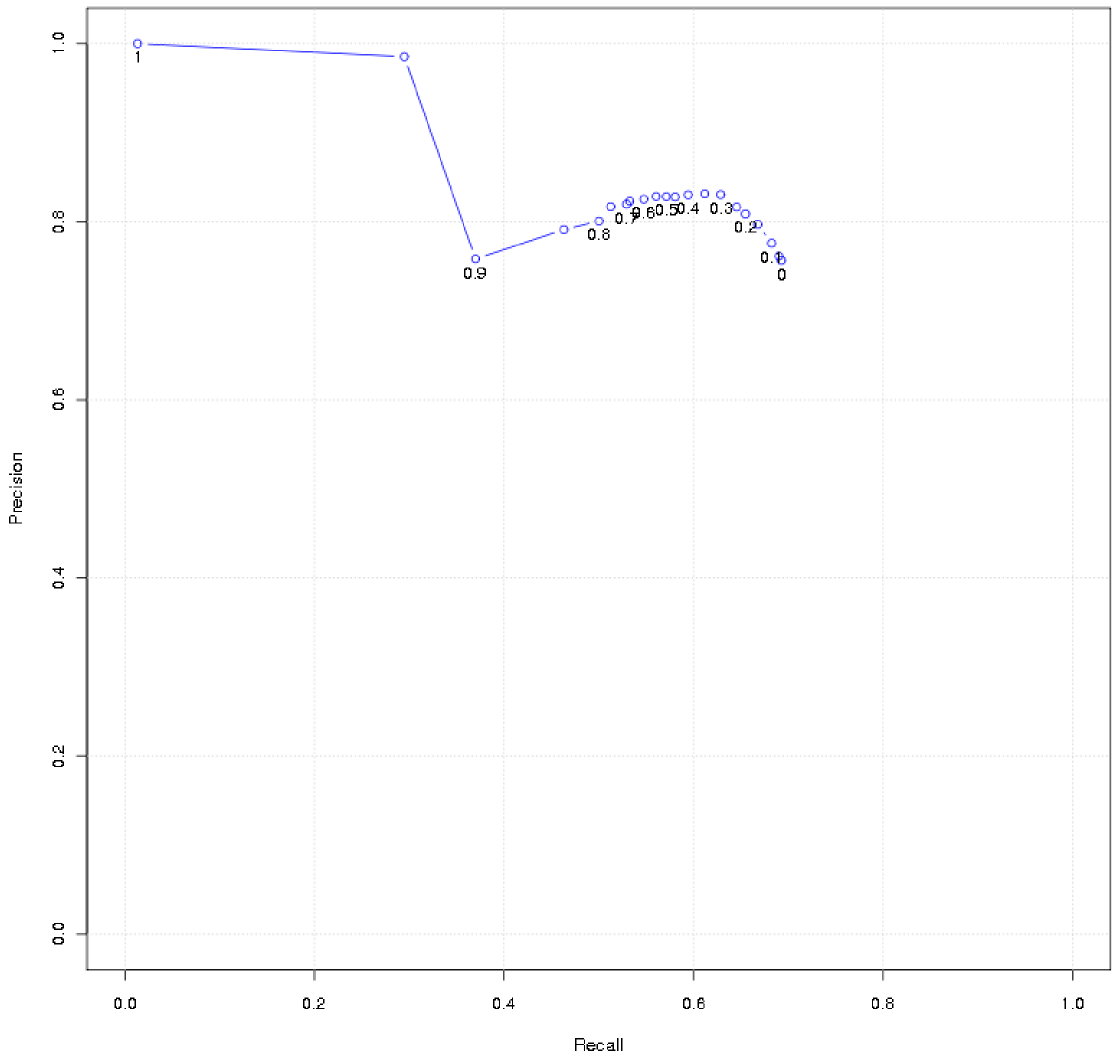


Figure 5.22: Precision-recall curve for OAEI test cases

Table 5.7: p-value from Wilcox test for 19 benchmark test cases. The green color means that OMNN is significantly better than the system; red color means the system is significantly better than OMNN; yellow means no significant difference. Significance is defined as p-value< 0.05.

System	Precision	Recall	F-Measure
aflood	🟥 0.000	🟨 0.570	🟨 0.182
AgrMaker	🟥 0.014	🟩 0.000	🟩 0.000
aroma	🟨 0.420	🟩 0.000	🟩 0.000
ASMOV	🟥 0.000	🟩 0.046	🟨 0.679
DSSim	🟥 0.027	🟩 0.000	🟩 0.000
GeRoMe	🟩 0.042	🟩 0.000	🟩 0.000
kosimap	🟥 0.008	🟩 0.000	🟩 0.000
Lily	🟥 0.000	🟨 0.306	🟥 0.000
MapPSO	🟩 0.000	🟩 0.000	🟩 0.000
RiMOM	🟨 0.136	🟩 0.002	🟩 0.032
SOBOM	🟨 0.811	🟩 0.000	🟩 0.000
TaxoMap	🟩 0.011	🟩 0.000	🟩 0.000

Table 5.8: Recall comparison between OMNN and preliminary result from textual information

test case	system	no training	20% training	40% training	60% training	80% training
249	OMNN	0.63	0.8	0.84	0.92	0.97
	preliminary	0.05	0.24	0.41	0.6	0.79
257	OMNN	0.27	0.67	0.79	0.94	1
	preliminary	0.03	0.21	0.42	0.61	0.79
248	OMNN	0.47	0.69	0.81	0.88	0.94
	preliminary	0.04	0.25	0.41	0.6	0.8
259	OMNN	0.4				0.95
	preliminary	0.02				0.79
265	OMNN	0.28				
	preliminary	0.03				
266	OMNN	0.21				
	preliminary	0.03				

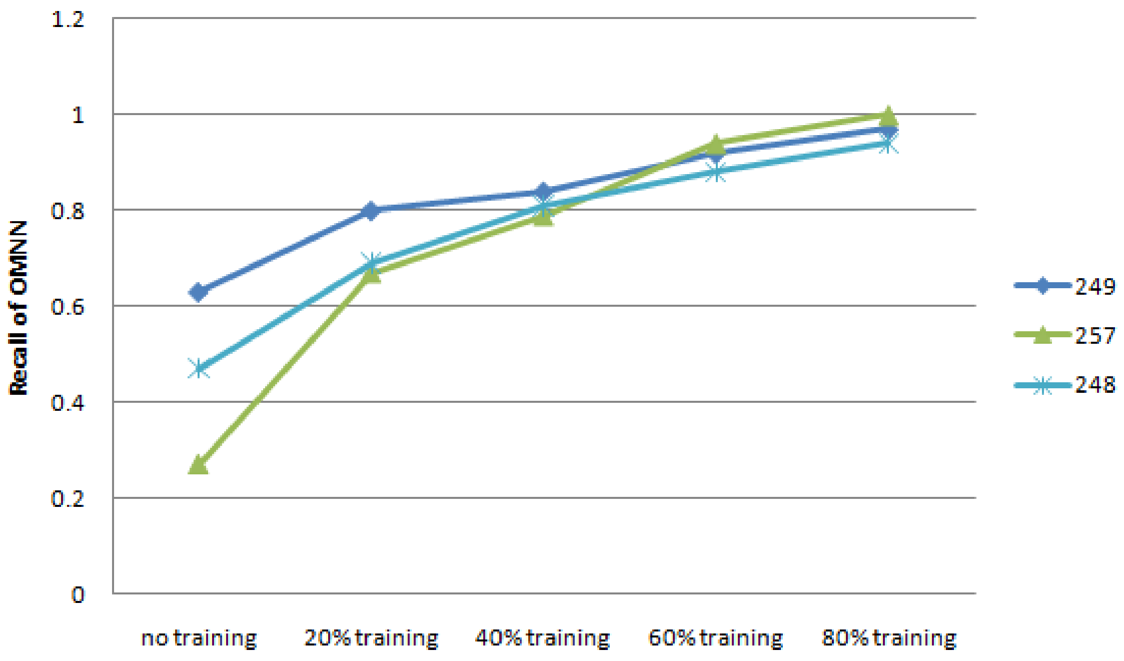


Figure 5.23: Recall of OMNN by different percentage of training data.

6.0 CONCLUSION AND FUTURE WORK

OMNN approach for graph mapping and ontology mapping is presented in this dissertation. The experiment results show that the OMNN is a generic and scalable graph mapping and ontology mapping approach. It is competitive with all the top-ranked systems on benchmark tests at OAEI ontology matching campaign 2009.

In this chapter, the main contributions of this dissertation are outlined and a number of directions of future work are presented.

6.1 SUMMARY OF CONTRIBUTIONS

This dissertation proposes a new neural network architecture, i.e., OMNN, which supports not only item mapping, but also relationship representation and mapping that is not available in IENN. A novel explicit training method for relationship mapping is proposed by backpropagating errors from representatin layer only.

Furthermore a new ontology mapping approach with a novel way of combining textual information and structural information is proposed to solve ontology mapping problem. The general idea of this approach is this: it first generates high confident mapping pairs with textual information, and then uses these pairs as cross-training data for OMNN to generate more mappings.

This dissertation applies the OMNN approach to various graph mapping test cases. Experimental results show that the network is able to fill in missing mappings and generate ambiguous results when ambiguities exist in mapping.

The OMNN ontology mapping approach is tested on 19 benchmark test cases from OAEI campaign 2009, and compared with other top ranked systems from OAEI 2009. Experimental results show OMNN approach is competitive to other systems.

At the heart of this dissertation is the conjecture that the capacity for analogical reasoning is supported by pathways shared for a variety of cognitive tasks that share features at an abstract level. By including (at least) two layers of hidden units, the proposed architecture includes (at least) one shared pathway of synaptic connections. It is hypothesized that these shared weights encode the high level features required to infer mappings that exhibit properties consistent with structural analogy. When a connectionist network is trained (for example, with backprop), the connection strengths encode statistical properties of the joint distribution of the pattern pairs in the training environment.

The OMNN approach adds further evidence onto the conjecture that the neural processes underlying analogical processing make use of overlapping pathways.

6.2 FUTURE DIRECTIONS

Given our findings, there is still work to be done.

Current OMNN approach supports 1-to-1 mapping only. However, it's possible that, in some cases, one item in one graph can be mapped to multiple items in another graph.

Actually OMNN's ability to 1-to-n mapping has been demonstrated in 4.4. To extend it to 1-to-n or even m-to-n mappings, the only thing we need to do is to improve the mapping extraction algorithm.

OMNN approach focuses on utilizing structural information. Even though any textual information could be used in generating training data, currently only the simple string matching algorithm is used. Since this simple edit distance based method can not deal with synonyms and cross-lingual mapping, more complex method utilizing textual information should be explored in future.

In real world ontology, some detailed information is not represented in OMNN approach. For example, in OWL language object property could be transitive and relationship could be between not only concepts but also properties, i.e. one property could be "subPropertyOf" another property. In current OMNN approach, this detailed information is not used. Future work should explore ways to represent them in OMNN network.

APPENDIX

OAEI 2009 BENCHMARK TESTS

The table below summarize what has been retracted from the reference ontology. There are 6 categories of alteration:

1. Name

   Name of entities can be replaced by (R/N) random strings, (S)ynonyms, (N)ame with different conventions, (F) strings in another language than english.

2. Comments

   Comments can be (N) suppressed or (F) translated in another language.

3. Specialization Hierarchy:

   Specialization Hierarchy can be (N) suppressed, (E)xpansed or (F)lattened.

4. Instances

   Instances can be (N) suppressed

5. Properties

   Properties can be (N) suppressed or (R) having the restrictions on classes discarded.

6. Classes

Classes can be (E)xpanded, i.e., relaced by several classes or (F)latened.

Table A1: Overview of OAEI 2009 benchmark tests

Number	Name	Comments	Specialization Hierarchy	Instances	Properties	Classes	Comments
101	0	0	0	0	0	0	Reference alignment
102							Irrelevant ontology
103	0	0	0	0	0	0	Language generalization
104	0	0	0	0	0	0	Language restriction
201	R	0	0	0	0	0	No names
202	R	S	0	0	0	0	No names& no comments
203	0	S	0	0	0	0	No comments (was misspelling)
204	C	0	0	0	0	0	Naming conventions
205	S	0	0	0	0	0	Synonyms
206	F	T	0	0	0	0	Translation
207	F	0	0	0	0	0	
208	C	S	0	0	0	0	
209	S	S	0	0	0	0	
210	F	S	0	0	0	0	
221	0	0	S	0	0	0	No specialization

Continued on next page

Number	Name	Comments	Specialization Hierarchy	Instances	Properties	Classes	Comments
222	0	0	F	0	0	0	Flattened hierarchy
223	0	0	E	0	0	0	Expanded hierarchy
224	0	0	0	S	0	0	No instance
225	0	0	0	0	R	0	No restrictions
228	0	0	0	0	N	0	No properties
230	0	0	0	0	0	F	Flattened classes
232	0	0	N	N	0	0	
233	0	0	N	0	N	0	
236	0	0	0	N	N	0	
237	0	0	F	N	0	0	
238	0	0	E	N	0	0	
239	0	0	F	0	N	0	
240	0	0	E	0	N	0	
241	0	0	N	N	N	0	
246	0	0	F	N	N	0	
247	0	0	E	N	N	0	
248	N	N	N	0	0	0	
249	N	N	0	N	0	0	

Continued on next page

Number	Name	Comments	Specialization Hierarchy	Instances	Properties	Classes	Comments
250	N	N	0	0	N	0	Individual is empty
251	N	N	0	0	N	0	
252	N	N	E	0	0	0	
253	N	N	N	N	0	0	
254	N	N	N	0	N	0	
257	N	N	0	N	N	0	
258	N	N	F	N	0	0	
259	N	N	E	N	0	0	
260	N	N	F	0	N	0	
261	N	N	E	0	N	0	
262	N	N	N	N	N	0	
265	N	N	F	N	N	0	
266	N	N	E	N	N	0	
301							Real: BibTeX/MIT
302							Real: BibTeX/UMBC
303							Real: Karlsruhe
304							Real: INRIA

BIBLIOGRAPHY

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: Scientific american. *Scientific American*, May 2001.

[BM06] Jianghua Bao and Paul W. Munro. Structural mapping with identical elements neural network. In *Proceedings of the International Joint Conference on Neural Networks - IJCNN 2006*, pages 870–874. IEEE, 2006.

[CGL01] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Ontology of integration and integration of ontologies. In *Description Logics*, 2001.

[CM03] M. Crubezy and M. Musen. Ontologies in support of problem solving, 2003.

[DMDH02] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web, 2002.

[DMDH03] An-Hai Doan, J. Madhavan, Pedro Domingos, and Alon Halevy. Learning to map ontologies on the semantic web. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 662–673, 2003.

[DMQ03] Dejing Dou, Drew Mcdermott, and Peishen Qi. Ontology translation on the semantic web. In *Journal of Data Semantics*, page 2005, 2003.

[DMQ05] D. Dou, D. McDermott, and P. Qi. Ontology translation on the Semantic Web. *Journal on Data Semantics (JoDS)*, II:35–57, 2005.

[EBB$^+$04] J. Euzenat, J. Barrasa, P. Bouquet, R. Dieng, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. van Acker, I. Zaihrayeu, and T. L. Bach. D2.2.3:

State of the art on ontology alignment. Technical report, NoE Knowledge Web project delivable, 2004. http://knowledgeweb.semanticweb.org/.

[Ehr06] Marc Ehrig. *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[ES04] Marc Ehrig and Steffen Staab. QOM: Quick ontology mapping. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, pages 683–697, 2004.

[ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[GH05] D. Gasevic and M. Hatala. Ontology mappings to improve learning resource search. *Journal of Educational Technology(Special issue on Advances of Semantic Web for E-learning: Expanding learning frontiers).*, 2005.

[GKD97] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *in proceedings of 1997 ACM SIGMOD Conference*, pages 539–542, 1997.

[Gru93a] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–221, 1993.

[Gru93b] Tom Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.

[HA09] Md. Seddiqui Hanif and Masaki Aono. Anchor-flood: Results for oaei 2009. In Shvaiko et al. [SEG$^+$09].

[Hin86] G. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, 1986.

[HIST03] Alon Halevy, Zachary Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, 2003.

[JMSK09] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. Asmov: Results for oaei 2009. In Shvaiko et al. [SEG+09].

[MB05] P. Munro and J. Bao. A connectionist implementation of identical elements. In *In Proceedings of the Twenty Seventh Ann. Conf. Cognitive Science Society Proceedings*. Lawerence Erlbaum: Mahwah NJ, 2005.

[MGMR02] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE), San Jose (CA US)*, pages 117–128, 2002.

[MKSI96] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 14–25, 1996.

[MP09] Paul Munro and Yefei Peng. Analogical learning and inference in overlapping networks. In B. Kokinov, K. Holyoak, and D Gentner, editors, *New Frontiers in Analogy Research*. New Bulgarian University Press, 2009.

[MS07] C. Meilicke and H. Stuckenschmidt. Analyzing mapping extraction approaches. In *In Proc. of the ISWC 2007 Workshop on Ontology Matching, Busan, Korea*, 2007.

[Mun96] P. Munro. Shared network resources and shared task properties. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, 1996.

[Mun08] Paul W. Munro. Learning structurally analogous tasks. In Vera Kurková, Roman Neruda, and Jan Koutník, editors, *Artificial Neural Networks - ICANN 2008, 18th International Conference*, volume 5164 of *Lecture Notes in Computer Science*, pages 406–412. Springer: Berlin/Heidelberg, 2008.

[NDH05] Natalya F. Noy, AnHai Doan, and Alon Y. Halevy. Semantic integration. *AI Mag.*, 26(1):7–9, 2005.

[NM00] Natasha Noy and Mark Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th AAAI, Austin (TX US)*, pages 450–455, 2000. http://citeseer.nj.nec.com/528663.html.

[NM01] Natasha Noy and Mark Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US)*, pages 63–70, 2001. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-47/.

[NM03] N. F. Noy and M. A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

[Noy04] N. Noy. Semantic Integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[PM09] Yefei Peng and Paul Munro. Learning mappings with neural network. In *Proceedings of the 2009 International Conference on Artificial Intelligence*, 2009.

[RB01] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[SEG$^+$09] Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natalya Fridman Noy, and Arnon Rosenthal, editors. *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly, USA, October 25, 2009*, volume 551 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[UG04] M. Uschold and M. Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[vEdBvdHM01] Rogier van Eijk, Frank de Boer, Wiebe van de Hoek, and John-Jules Meyer. On dynamically generated ontology translators in agent communication. *International journal of intelligent system*, 16:587–607, 2001.

[WX09] Peng Wang and Baowen Xu. Lily: Ontology alignment results for oaei 2009. In Shvaiko et al. [SEG+09].

[ZZS+09] Xiao Zhang, Qian Zhong, Feng Shi, Juanzi Li, and Jie Tang. Rimom results for oaei 2009. In Shvaiko et al. [SEG+09].