Finding Best Probable Combinations of Similarity Measures Using Bayesian Networks

[1]Kalpana Nigam, [2]Monica Mehrotra
[1]Maharaja Agrasen College, University of Delhi, New Delhi, India
[2]Dept. of Comp.Sc. Jamia Millia Islamia, New Delhi, India

Abstract
The objective of establishing alignments between two entities from two different ontologies is focused on finding similarity between two strings. Out of several available similarity measures it becomes difficult to select best one or best combination. In this paper we explore the use of Bayesian Networks to aid decision making under uncertainty. We present here a method to find out the best possible combination of similarity measures which can be taken for aggregation for the computation of final similarity value of two strings by learning Bayesian Networks.

Keywords
Ontology Alignment, String Similarity, Bayesian Networks Learning

I. Introduction

Achieving semantic interoperability among different information systems is very laborious, tedious and error-prone in a distributed and heterogeneous environment like the World Wide Web (WWW). The Semantic Web relies heavily on the formal ontologies that structure underlying data for the purpose of comprehensive and transportable machine understanding [12]. Therefore, the success of the Semantic Web depends strongly on the proliferation of ontologies, which requires fast and easy engineering of ontologies and avoidance of a knowledge acquisition bottleneck. Ontology [5, 6] provides a shared vocabulary so that two agents can understand what is communicated. However by design, web is decentralized and heterogeneous. This means, it is highly unrealistic to expect to have a single Ontology that all parties agreed upon. On the other hand ontologies themselves may have heterogeneities as well i.e. many entities in different ontologies may refer to a single concept. To overcome this problem of heterogeneity, ontology alignment seems to be a solution to provide interoperability to the semantic web. In our endeavor of enhancing ontology alignment, we mainly remained focused on the most important step of the alignment i.e. measuring similarity between the entities of two different ontologies of similar domain. We have combined different individual similarity metrics of string-based, linguistic, structural categories and instance based into one input sample. As each individual similarity measure is able to determine partial similarity of the whole feature space, considering all the measures simultaneously will probably achieve higher classification accuracy. The ensemble method is an active research area which gives better performance than a single measure [14].
To find the best combination of similarity measures, in this paper we explore the use of Bayesian Networks learning, which gives a hint of best probable combination of similarity measures required in ontology alignment.
Before presenting our method, we present a brief introduction on Bayesian Networks in the next section.

II. Bayesian Networks

A Bayesian belief network, usually just called a Bayesian network, is a graphical tool that can aid decision-making under uncertainty [3, 17]. The networks represent a system over which a probability distribution is defined, modeling uncertainty both quantitatively and qualitatively. They allow a user to make inferences when only limited information is available. Mathematically, a Bayesian network is a directed acyclic graph whose nodes represent variables [2]. A link from one node to another represents a causal dependency. The network is used to analyze uncertain information and draw conclusions from the data.
Bayesian networks use probability to represent uncertainty. A probability distribution indicates the strength of our belief in uncertain information or inferences. Each variable or node in the network consists of a finite set of mutually exclusive states. The directed links between variables in the graph represent causal relationships. A link from variable or node B to variable or node A indicates that B can cause A. We say B is a parent of A, and A is a child of B. Because the network is acyclic, causal feedback is not an issue. Each variable has a probability table associated with it. Variables with no parents have a very simple probability table, giving the initial probability distribution of the variable. Variables with parents are much more complicated. These variables have conditional probability tables, which give a probability distribution for every combination of states of the variable's parents. A Formal definition of a Bayesian network as found in [2] is presented below:
Definition A Bayesian network is a pair (G, P), where G = (V, E) is a directed acyclic graph (DAG) over a finite set of nodes (or vertices), V, interconnected by directed links (or edges), E, and P is a set of (conditional) probability distributions. The network has the following property:

- Each node representing a variable A with parent nodes representing variables B_1, B_2,..., B_n (i.e., Bi A for each i=1,...,n) is assigned a Conditional Probability Table (CPT) representing $P(A \mid B_1, B_2, ..., B_n)$.

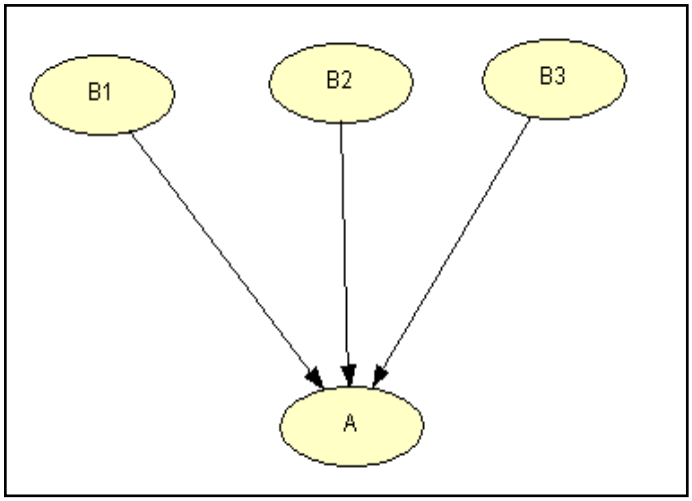

Fig. 1: Bayesian Network in Which Events B_1, B_2 and B_3 Affect the Event A

The nodes represent random variables, and the links represent probabilistic dependencies between variables. These dependencies

are quantified through a set of Conditional Probability Tables (CPTs): Each variable is assigned a CPT of the variable given its parents. For variables without parents, this is an unconditional (also called a marginal) distribution

A. Creating and Using Bayesian Networks

Building a Network includes two main steps:

1. Constructing the graphical model of nodes and links
2. Constructing probability tables for each node in the network [2].

1. Constructing the Graphical Model

To construct a Bayesian network, we must first determine the hypothesis variables. These are the variables for which we have to determine the probability distribution, and which answer our query. Secondly, evidence variables (sometimes called information variables) are added. These represent things that can be observed about the system being modelled. When events are observed, these variables allow the information to be entered into the network. Finally, intermediate, or mediating, variables are added. These variables may not be necessary, as they generally provide no extra information. However, they are useful as they allow the network structure to accurately represent the modelled system. This also dramatically reduces the size of the conditional probability tables. The network is formed by linking these variables using directed edges (arrows). It is vital to ensure that the links follow the direction specified by causality.

For example, consider two variables labeled 'Season' and 'Road Conditions'. The arrow points from 'Season' to 'Road Conditions', because the Road Conditions casually depend on Season and similarly 'Number of causalities' casually depend on 'Number of Journeys'.

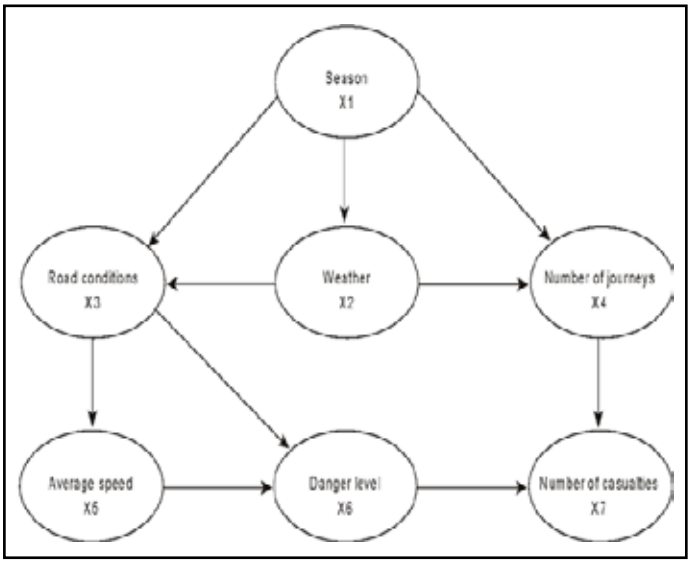

Fig. 2: An Example of Bayesian Network from RITA (Research and Innovative Technology Administration Bureau of Transportation Statistics)

2. Constructing the Probability Tables

As the Bayes' Rule[2] suggests, the conditional probability of an event A assuming that B has occurred, denoted $P(A \mid B)$, equals

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}, \tag{1}$$

which can be proven directly using a Venn diagram. Multiplying through, this becomes

$$P(A \mid B)\,P(B) = P(A \cap B), \tag{2}$$

which can be generalized to

$$P(A \cap B \cap C) = P(A)\,P(B \mid A)\,P(C \mid A \cap B). \tag{3}$$

Rearranging (1) gives

$$P(B \mid A) = \frac{P(B \cap A)}{P(A)}. \tag{4}$$

Solving (4) for $P(B \cap A) = P(A \cap B)$ and plugging in to (1) gives

$$P(A \mid B) = \frac{P(A)\,P(B \mid A)}{P(B)}.$$

Once the network structure is completed, the next task is to fill the probability tables. For variables with no parents, this task is very simple, as an initial probability can usually be assigned to each state from data or information already known, consulting a subject matter expert if necessary. The probability tables of variables with parents are usually more complicated, as these tables can easily grow up to several hundred entries. Small tables can be filled directly, using subjective probabilities which can be provided by an expert, or by probabilities based on gathered data

Reducing the size of the probability tables can make the process of completing them much easier. This can be done in two ways; by reducing the number of states of the variable or its parents, or by using intermediate variables to reduce the number of parents that a variable has. The first method is often impractical, but should be kept in mind. The second method is the most useful, as intermediate variables can be inserted in most areas of a network.

III. Finding Best Combination of Similarity Measures Using BNs

A. Constructing Bayesian Network of Similarity Measures

As discussed in previous chapters, our focus is on similarity measures for enhancing the ontology alignment. In this process of establishing ontology alignments, we have considered all four categories of measures (such as string-based, language-based, structure-based and instance-based) for finding similarity between the entities of two different ontologies. This approach of us is based on the concept of dependencies of different methods on each other, which makes it a meaningful target for BN modeling.

1. Experimental Setup

We conducted our experiment to find the best combination of measures from ten string-based methods and three language-based measures, an extension to[16]. For the string-based methods, the similarity values were computed by our program in C and for the linguistic measures, we computed the similarity values using WordNet::Similarity [13]. We have extracted 878 potential candidate entities (classes and relations) from more than 15 ontologies belonging to the bibliographic domain. The objective was to construct a Bayesian Network of similarity methods used for establishing alignments between the entities (classes and relations) of two ontologies from same domain.

Our idea was to overcome inefficiencies of several diverse types of mapping methods, by combining only those which are more significant or effective than the others, using Bayesian

Networks. We constructed the Bayesian Network in which each node represented a similarity method (string-based and semantic-based). The causal dependencies among the methods were also considered as we represented a method as parent node and it's variation as its child node. Such as SMOA[15] is based on Jaro-Winkler[18]. Jaro-Winkler is a variation to the Jaro algorithm[9], Dice Coefficient[4] and Jaccard similarity methods[9] are a variation of N-Gram[47] and Needleman-Wunsch[11] is a variation of Levenshtein algorithm[10] etc.

One additional node 'Align' in the network represented real alignments which causes all the happening of the similarity methods. The dependencies of the nodes represented their statistical inter-relationship. We created a Baysian Network consisting of ten string-based measures and another with four linguistic-based measures to understand their interdependencies.

We implemented our problem using HUGIN tool (http:// www.hugin.com)[1]. HUGIN EXPERT A/S has a high focus on delivering advanced solutions for decision making under uncertainty in the financial service industry. The HUGIN Decision Engine (HDE) implements state-of-the-art algorithms for Bayesian networks and influence diagrams such as object-oriented modeling, learning from data with both continuous and discrete variables, value of information analysis, sensitivity analysis and data conflict analysis.

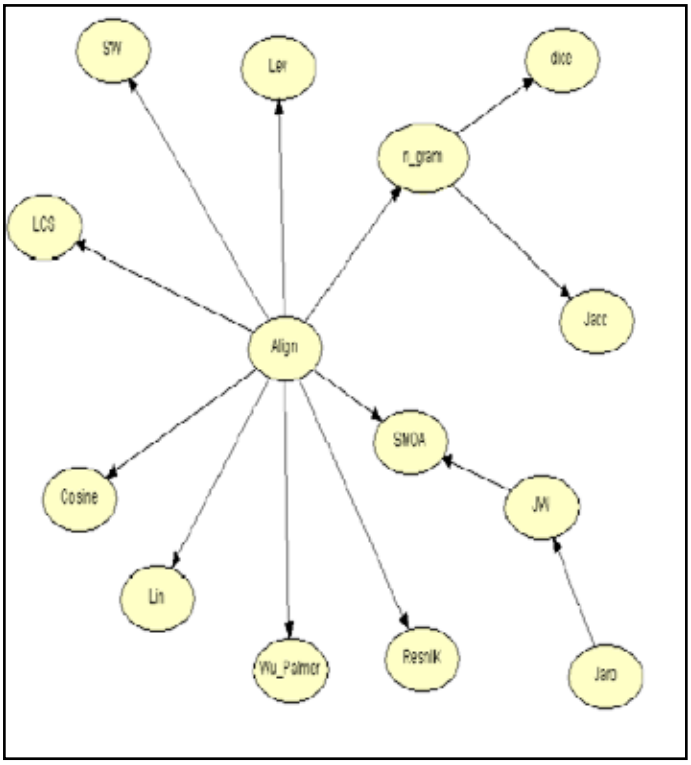

Fig. 3: BN Structure

After the Network is constructed, we populated Conditional Probability Tables (CPT) of each node as the average accuracy computed for each method in our previous experiments. The accuracy computation of the similarity measures is shown in the chapter three. The following Figure 4 shows the conditional probability table constructed for each node.

The CPTs for the nodes with no parent were filled directly by taking its accuracy. For the nodes which have parents, conditional probabilities were computed using Bayes's rule. After the compilation, the network showed 88% accuracy for the 'Align' node, which may be interpreted as the combined result of ten measures.

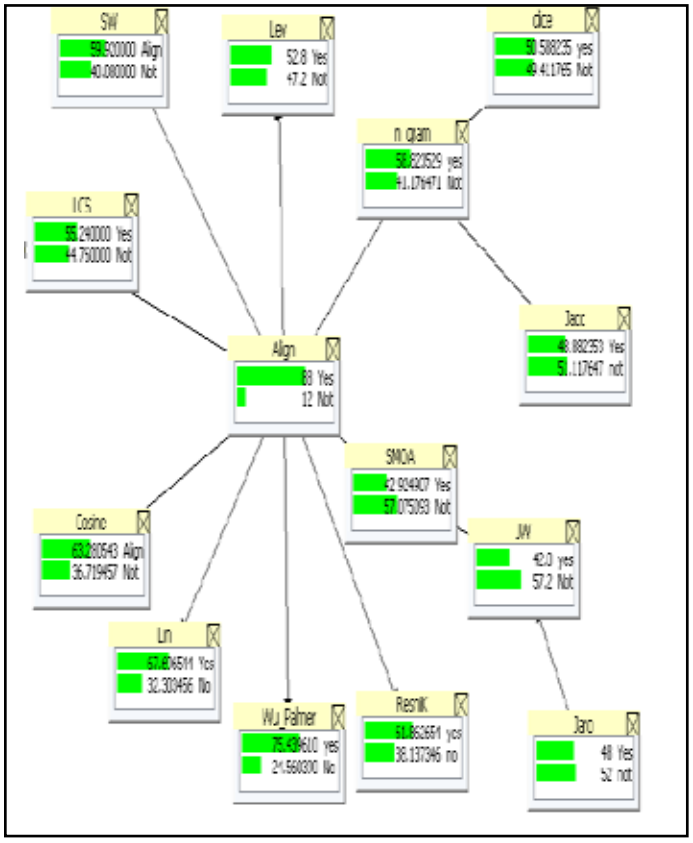

Fig. 4: Network With Probability Tables

When the evidence is entered into the system by selecting one of the states of information variable, or if necessary, by entering a probability distribution, the network's probabilities are then updated to reflect the new information. If more information is received, or some of the current evidences change, then the network is simply modified and recalculated. In our experiment we see a small change in the probability values when we entered the evidence that Aligned values are 100% true.

B. Learning Bayesian Networks

In the next step we train the network to understand the significance or the effective importance of the measures on the result. For this we have applied training and learning the Bayesian Network. Learning is the process of creating a Bayesian network from prior knowledge of the system[8]. The prior knowledge is usually in the form of large amounts of statistical data, but can also include network fragments, probabilities and causal relationships. Using these learning techniques means that a Bayesian network can be constructed with significantly less effort than if it was done by hand. With enough data, the resulting network can also be more accurate than the hand-made network. Structural learning, or the creation of the networks structure, relies on thousands of cases each giving the state of all of the networks variables at a particular point in time. If this information is required be accurately generated from gathered data in suitable quantities, the resulting learned structure will be nearly as accurate as a conventional hand-made network, and can sometimes even point out mistakes in the conventional network's structure.

Learning is available in one of the software programs named as Hugin [1], although there are restrictions on what can be done. It is an area of much research, and to use the full power of new techniques custom-built software will probably be required. Learning is only a useful technique where there are large amounts of accurate data available. (Of course the accuracy of the resulting

network depends on the accuracy of the simulation.) Applications of learning include data analysis and classification/recognition problems.

There are two constraint-based algorithms available for structure learning[1, 8]: The PC-Path Condition algorithm and the NPC-Necessary Path Condition algorithm. The basic idea of these algorithms is to derive a set of conditional independence and dependence statements (CIDs) by statistical tests. The algorithm performs the following steps:

- Statistical tests for conditional independence are performed for all pairs of variables).
- An undirected link is added between each pair of variables for which no conditional independences were found. The resulting undirected graph is referred to as the skeleton of the learned structure.
- Colliders are then identified, ensuring that no directed cycles occur. (A collider is a pair of links directed such that they meet in a node.) For example, if we find that A and B are dependent, B and C are dependent, but A and C are conditionally independent given S, not containing B, then this can be represented by the structure A --> B <-- C.
- Next, directions are enforced for those links whose direction can be derived from the conditional independences found and the colliders identified.
- Finally, the remaining undirected links are directed randomly, ensuring that no directed cycles occur.

One important thing to note about the PC algorithm is that, in general, it will not be able to derive the direction of all the links from data, and thus some links will be directed randomly. This means that the learned structure should be inspected, and if any links seem counterintuitive (e.g., sweat causes fever, instead of the other way around), one might consider using the NPC algorithm, which allows the user to interactively decide on the directionality of undirected links.

Traditional constraint-based learning algorithms produce provably correct structures under the assumptions of infinite data sets, perfect tests, and DAG faithfulness (i.e., that the data can be assumed to be simulated from a probability distribution that factorizes according to a DAG). In the case of limited data sets, however, these algorithms often derive too many conditional independence statements. Also, they may in some cases leave out important dependence relations.

Generally, it is recommended to use the NPC algorithm, as the resulting graph will be a better map of the (conditional) independence relations represented in the data. In particular, when the data set is small, the NPC algorithm should be the one preferred. The NPC algorithm, however, has longer running times than the PC algorithm.

The NPC algorithm, developed by researchers at Siemens in Munich, seeks to repair the deficiencies of the PC algorithm, which occur especially in the face of limited data sets. The solution provided by the NPC algorithm is based on inclusion of a criterion known as the 'necessary path condition'. This criterion forms the basis for introducing the notion of ambiguous regions, which in turn provide a language for selecting among sets of inter-dependent uncertain links. The resolution of ambiguous regions is performed in interaction with the user.

In constraint-based learning algorithms, the skeleton of the graph is constructed by not including a link in the induced graph whenever the corresponding nodes are found to be conditional independent. There can, however, be inconsistencies among the set of conditional independence and dependence statements (CIDs) derived from limited data sets. That is, not all CIDs can be represented simultaneously. The inconsistencies are assumed to stem solely from sampling noise; i.e., we still assume that there exists a perfect independence map of the (unknown) probability distribution from which the data is generated. The number of inconsistencies in the set of CIDs reflects structural model uncertainty.

Thus, the number of uncertainties is a confidence measure for the learned structure and can as such be used as an indication of whether or not sufficient data has been used to perform the learning. The inconsistent CIDs produce multiple solutions when inducing a Directed Acyclic Graph (DAG) from them. These solutions differ with respect to the set of links included. To resolve the inconsistencies, the NPC algorithm relies on user interaction where the user gets the opportunity to decide on directionality of undirected links and to resolve the ambiguous regions.

We extracted candidate entities from bibliographic ontologies and computed similarities between them. Our training set consists of 1080 pairs, out of which 782 were manually labeled as positive and 298 as negative. The results were transformed from the [0, 1] scale to two categories 'Aligned' if the value was over 0.5 and 'Not' if the value was less than 0.5 each.

The following fig. 5 shows the few lines from the data file, which we created for learning. The input to the process of BN training for ontology mapping is positive and negative examples with results of individual methods. To Learn the BN we used the Hugin Tool (http://www.hugin.com), the structure was trained using the NPC method. The figure 6 shows few lines from our simulated cases where "align/Align/yes" simulates a similarity value>=0.5 and "not/Not" a similarity value<0.5.

Lev	Sw	Jw	Align	Jaro	SMOA	n_gram	dice	Jacc	LCS	Cosine
0.57142	0.92307	0.74603	1	0.5	0.92307	0.74603	0.57142	0.92307	0.57142	0.95883
0.88888	0.94444	0.92592	1	0.88888	0.94117	0.94444	0.88888	0.94444	0.78571	0.94444
1	1	1	1	1	1	1	1	1	1	1
0.66666	0.94117	0.80555	1	0.63636	0.94444	0.80555	0.63636	0.94117	0.66666	0.94117
0.5	0.76923	0	1	0.5	0.76923	0	0.57142	0.72727	0.57142	0.76923
0.63636	0.8421	0.88846	1	0.63636	0.8421	0.88846	0.63636	0.8421	0.63636	0.88888
0.42857	0.72727	0	1	0.42857	0.72727	0	0.42857	0.72727	0.42857	0.72727
0.6	0.82352	0.93928	1	0.6	0.82352	0.93928	0.6	0.82352	0.6	0.82352
0.63636	0.8421	1	1	0.63636	0.8421	1	0.63636	0.8421	0.63636	0.8421
0.42857	0.72727	0.46428	1	0.42857	0.72727	0.46428	0.42857	0.72727	0.42857	0.76723
0.78571	0.88	0	1	0.78571	0.88	0	0.78571	0.8421	0.78571	0.88
0.3	0.54736	0.74814	1	0.3	0.54736	0.74814	0.3	0.54736	0.3	0.54736
0.5	0.76923	0	1	0.5	0.76923	0	0.5	0.72727	0.57142	0.76923
0.3	0.57142	0.65333	1	0.3	0.57142	0.65333	0.38888	0.57142	0.33333	0.57142
0.78571	0.88	0.27489	1	0.78571	0.88	0.33333	0.78571	0.88	0.78571	0.88
0.72727	0.8421	1	1	0.72727	0.8421	1	0.72727	0.8421	0.72727	0.8421
0.5	0.76923	0.55	1	0.5	0.76923	0.55	0.50	0.76923	0.5	0.76923
0	0.72727	0.46428	0	0	0.72727	0.46428	0	0.72727	0	0.72727
0	0.95883	0.61111	1	0	0.95883	0.61111	0	0.95883	0	0.95883
0.11	0.8	0.42	0	0.11	0.8	0.42	0.3	0.8	0.33333	0.8
0.42857	0.72727	0	1	0.42857	0.72727	0	0.42857	0.72727	0.42857	0.7
0.6	0.82352	0.93928	1	0.6	0.82352	0.93928	0.6	0.82352	0.6	0.82352
0.63636	0.8421	1	1	0.63636	0.8421	1	0.63636	0.82352	0.63636	0.8421
0.42857	0.72727	0.46428	1	0.42857	0.72727	0.46428	0.42888	0.72727	0.42857	0.76923
0.78571	0.88	0	0	0.78571	0.88	0	0.78571	0.88	0.78571	0.88
0.3	0.54736	0.74814	0	0.3	0.54736	0.74814	0.3	0.54736	0.3	0.54736
0.5	0.76923	0	0	0.57142	0.76923	0	0.5	0.76923	0.5	0.76923
0.3	0.57142	0.65333	0	0.33333	0.57142	0.65333	0.3	0.57142	0.3	0.57142
0.78571	0.88	0.27489	0	0.78571	0.88	0.27489	0.78571	0.88	0.78571	0.88
0.72727	0.8421	1	0	0.72727	0.8421	1	0.72727	0.8421	0.72727	0.8421
0.5	0.76923	0.55	0	0.5	0.76923	0.55	0.5	0.76923	0.5	0.76923
0	0.72727	0.46428	0	0	0.72727	0.46428	0	0.72727	0	0.72134
0	0.95883	0.61111	0	0	0.95883	0.61111	0	0.95883	0	0.95883
0.11	0.8	0.42	0	0.11111	0.8	0.42	0.11	0.8	0.11	0.8888
0.42857	0.72727	0	0	0.42857	0.72727	0	0.42857	0.72727	0.42857	0.72727
0.6	0.82352	0.93928	1	0.6	0.82352	0.93928	0.6	0.82352	0.6	0.82352
0.63636	0.8421	1	0	0.63636	0.8421	1	0.63636	0.8421	0.63636	0.8421
0.42857	0.72727	0.46428	0	0.42857	0.72727	0.46428	0.42857	0.72727	0.42857	0.72727
0.78571	0.88	0	0	0.78571	0.88	0	0.78571	0.88	0.78571	0.88
0.3	0.54736	0.74814	0	0.3	0.54736	0.74814	0.3	0.54736	0.3	0.54736
0.5	0.76923	0	0	0.5	0.76923	0	0.5	0.76923	0.5	0.76923
0.3	0.57142	0.65333	0	0.3	0.57142	0.65333	0.3	0.57142	0.3	0.57142
0.78571	0.88	0.27489	0	0.78571	0.88	0.27489	0.78571	0.88	0.78571	0.88
0.72727	0.8421	1	0	0.72727	0.8421	1	0.72727	0.8421	0.7	0.8
0.5	0.76923	0.55	0	0.5	0.76923	0.55	0.5	0.76923	0.5	0.76923
0	0.72727	0.46428	0	0	0.72727	0.46428	0	0.76923	0	0.72727
0	0.95883	0.61111	1	0	0.95883	0.61111	0	0.95883	0	0.95883
0.11	0.8	0.42	1	0.3	0.8	0.42	0.11	0.8	0.11	0.8
0.42857	0.72727	0	1	0.42857	0.72727	0	0.42857	0.72727	0.42857	0.72727
0.6	0.82352	0.93928	1	0.6	0.82352	0.93928	0.6	0.82352	0.6	0.82352
0.63636	0.8421	1	1	0.63636	0.8421	1	0.63636	0.8421	0.63636	0.8421
0.42857	0.72727	0.46428	1	0.42857	0.76923	0.46428	0.42857	0.73455	0.42857	0.72727

Fig. 5: Few Lines from Data File

The fig. 7 shows the learnt structure. From the learnt structure, we conclude that if we know the mapping justifications of, Jaccard, SMOA, Levenstein, Jaro-Winkler and Wu & Palmer, other methods do not matter; because only for these methods, the leant structure shows clear links of dependency. Other similarity measures either not covered in the structure or only show a dependency among them, such as dice and n-gram. Also the structure shows one wrongly directed link between SMOA and Jaro-Winkler method.

Data

#	Lev	SW	JW	Align	Jacc	dice
0	align	align		yes		Align
1	not	not	align	not	not	Align
2	align	not	align	yes	Align	Not
3	align		align	not	not	Not
4	align	align	align	yes	not	Not
5	not	align	align	not	not	Not
6	not	not	align	not	Align	Not
7	align	not	align	yes	Align	Not
8	align	not		yes	Align	Not
9	align	align	align	not	not	Not
10	align	not	align	not	Align	Not
11	align	align	not	not	not	Align
12	not	align	not	yes	Align	Align
13	not	align	align	yes		Not
14	not	align	not	not	not	Not
15	not	align	align	not	not	Align
16	not	align	not	yes	Align	Align
17	align	align	not	not	not	Align
18		not	align	yes	Align	Align
19	not	align	align	yes	Align	Not
20	align		not	yes	not	Not
21	not	not	not	yes	Align	
22	align	not	align	yes	not	
23	not	align	align	yes	not	Align
24	align	not	not	yes	not	Not

Fig. 6: Few Lines After Simulation from the 1080 Cases

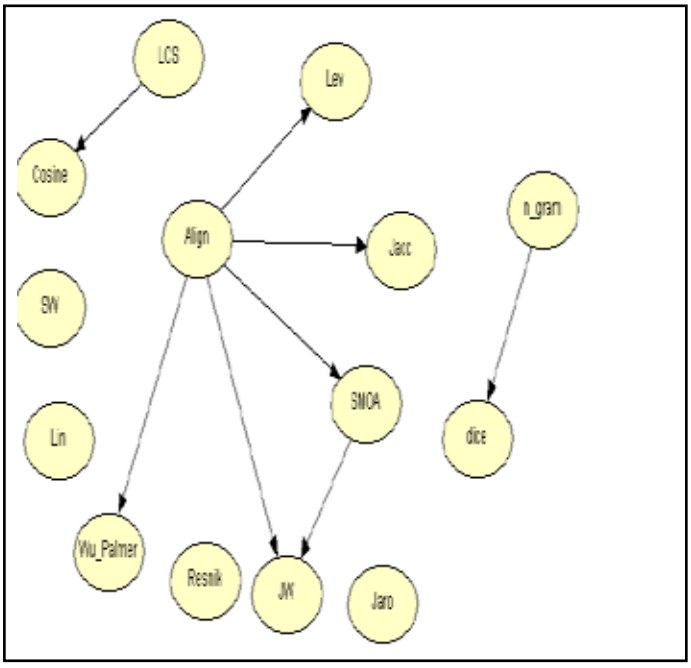

Fig. 7: Learnt Structure

We have observed during the repeated learning experiments that the learnt structure gives clearer picture as the training data set increases. In the initial phase of our experiment, the learnt structure was containing ambiguities and confusing structure, as the training data set for the linguistic measures was not large enough to portray a clear and meaningful structure. The addition of more samples in the training set was required for a clearer picture.

The results obtained after learning prompted us to check its correctness. For doing that we trained the neural network for the alignments, where similarity values obtained after the aggregation of the measures of this combination is taken as input in the Neural Network. The combination of five measures (Jaccard, SMOA, Levenshtein, Jaro-Winkler and Wu & Palmer) given by Bayesian Network learning, had accuracy between 81.14% and 59.94%. The repeated experiments showed that for a bigger learning data set, we get better results.

IV. Conclusion

We explored here the learning with Bayesian Networks, which suggested a particular combination of methods, from different string-based and language-based methods by training a Bayesian Network. We performed the learning with different datasets in different experiments. The learned structures suggest a strong correlation among some of the mapping methods but some of the results were found to be contradicting. The performance of these methods in combination is evaluated again by training a neural network to help us establish strong recommendations on the success of the combination of methods, found using Bayesian Networks learning. Our concern is to reduce the range of accuracy by improving our learning data set in the future work.

References

[1] [Online] Available: http://www.hugin.com
[2] Buede, D.M., Tatman, J.A., Bersnick, T.A.,"Introduction to Bayesian Networks", The 66th MORS Symposium, Naval Postgraduate School, Monterey, California, 23-25,1998.
[3] Charniak, E.,"Bayesian Networks without Tears", AI Magazine, winter 1991.
[4] Dice Lee, R.,"Measures of the Amount of Ecology Association between Species", Ecology 26(3), pp. 297-302, 1945.
[5] Gasevic, D., Hatala. M,"Ontology alignments to improve learning resource search", British Journal of Educational Technology (Special issue on Advances of Semantic Web for E-learning: Expanding learning frontiers), 2005.
[6] Giunchiglia, F., Shvaiko P.,"Semantic Matching", In Proceedings of IJCAI 2003, Workshop on ontologies and distributed systems, pp. 139– 146, 2003.
[7] Gusfield, D.,"Algorithm on Strings, Trees and Sequences", Computer science and Computational Biology, USA, Cambridge University Press.
[8] Heckerman, D.,"A Tutorial on Learning with Bayesian Networks", Technical report, Microsoft Research on Advanced technology Division, Microsoft Corporation, 1995.
[9] Jaccard, P.,"Étude comparative de la distribution florale dans une portion des Alpes et des Jura", Bulletin de la Société Vaudoise des Sciences Naturelles 37, pp. 547–579, 1901.
[10] Leacock, C., Chodorow, M.,"Combining Local Context And Wordnet Similarity For Word Sense Identification", In Fellbaum 1998, pp. 265-283, 1998.
[11] Navigli, R., Velardi, P.,"Ontology Learning and Its applications to Automated Terminology Translation", IEEE Intelligent Systems, 18(1), pp. 22-31, 2003.
[12] Noy, N.,"Semantic Integration: A Survey of Ontology-Based Approaches", SIGMOD Record 33(4), pp. 65-70, 2004.
[13] Pedersen, T., Patwardhan, S., Michellizzi, J., "WordNet: Similarity – Measuring the relatedness of concepts", In Proceedings of 19th National Conference on AI, pp. 1024-

1025, San Jose,C.A.,2008.
[14] Smith T.F., Waterman M.S.,"Identification of Common Molecular Subsequences", Molecular Biology, 147, 1981.
[15] Staab, S., Mädche A.,"Measuring Similarity Between Ontologies", Lecture Notes in Artificial Intelligence, 2473, pp. 251–263, 2002.
[16] Svab, O., Svatek, V.,"Combining Ontology Mapping Methods using Bayesian Networks", In: ISWC 2006, Athens, 2006.
[17] Tang, J., Li, J., Liang B., Huang, X., Li, Y., Wang, K.,"Using Bayesian Decision for Ontology Alignment", Web Semantics: Services and Agents on the World Wide Web, Vol. 4, pp. 243-262, December 2006.
[18] Wiederhold, G.,"Mediators in the architecture of future information systems", IEEE Computer 25(3), 1992.

Kalpana Nigam has received M.C.A. degree from Aligarh Muslim University, India and M.Phil. from Madurai Kamraj University, India. She registered as a Ph.D. student in Jamia Millia Islamia, India in 2008. She is currently working as Assistant Professor in Computer Science with Maharaja Agrasen College, University of Delhi, India. She has more than 16 years of teaching experience and successfully handles administrative work in Maharaja Agrasen College, University of Delhi. Her research interests include Ontology Alignment, Machine Learning Techniques, Data Structure and Database Management System.

Dr. Monica Mehrotra has been in the teaching profession for almost two decades now. She completed her MCA from IET, Lucknow University and Ph.D from Department of Computer Science, Jamia Millia Islamia, New Delhi. Her research has been in area of 'Intelligent Information retrieval'. She has been attached to the Department of Computer Science, Jamia Millia Islamia since Oct 2000 in the capacity of Asst. Prof. She has over 35 publications in International Conferences and International Journals. Current Research Areas are Data Mining, Information Retrieval, Ontologies, Software Security, and Big data analytics.