A Unified Probabilistic Approach for Semantic Clustering of Relational Phrases

Adam Grycner, Gerhard Weikum
Max-Planck Institute for Informatics
Campus E1.4, 66123
Saarbrücken, Germany
{agrycner, weikum}@mpi-inf.mpg.de

Jay Pujara, James Foulds, Lise Getoor
Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064
{jay, jfoulds, getoor}@soe.ucsc.edu

Abstract

The task of finding synonymous relational phrases is important in natural language understanding problems such as question answering and paraphrase detection. While this task has been addressed by many previous systems, each of these existing approaches is limited either in expressivity or in scalability. To address this challenge, we present a large-scale statistical relational method for clustering relational phrases using Probabilistic Soft Logic (PSL) [1]. To assess the quality of our approach, we evaluated it relative to a set of baseline methods. The proposed technique was found to outperform the baselines for both clustering and link prediction, and was shown to be scalable enough to be applied to 200,000 relational phrases.

1 Introduction

To fully understand a written text, a machine must be able to understand the meaning of the relational phrases occurring within it. Relational phrases are textual representations of relations which occur between common noun phrases (e.g., "the movie star") or named entities (e.g., "George Clooney") with the same *semantic type signature*. For example, "Bob is married to Alice" connects two entities of the type <*person*>, and is an instance of the relational phrase "<*person*> is married to <*person*>." In this case, both the left side (domain) and the right side (range) arguments of the phrase have the type <*person*>. The problem of discovering and organizing relational phrases was addressed in many previous works [2, 3, 4, 5].

A key step toward detecting the meaning of these phrases is to find their synonyms, in a task known as semantic clustering. In this problem the goal is to group together phrases which have similar meanings. For example, "<*person*> is married to <*person*>" should be clustered together with a relation "<*person*> is a spouse of <*person*>." This kind of clustering has many applications, including paraphrase detection or generation, information extraction, semantic parsing, and question answering.

State of the art systems for clustering relational phrases concentrate mostly on two aspects – syntax similarity and argument co-occurrence statistics. The PATTY system [4] performs clustering using both argument overlap statistics and phrase syntax similarity. WiseNet [5] uses similar ideas but includes a soft clustering option. Universal Schema [6] applies collaborative filtering on the relation-arguments co-occurrence matrix to find similarity scores between relational phrases. Moreover, it learns vector representations for phrases, which allows to encode asymmetry between them. Universal Schema additionally combines closed IE (prespecified) with open IE (newly discovered) phrases. NELL [7] also organizes extracted relations into groups of synonyms, but the number of clusters is limited to a fixed set of prespecified relations. In our case, the number of clusters is unknown. DIRT [8] was one of the first works which addressed finding synonymous relational

phrases using the distributional hypothesis [9]. RESOLVER [10] extended that idea with computing pairwise similarities between relations and applying hierarchical agglomerative clustering (HAC). Unlike PATTY, WiseNet or our PSL models, both of the aforementioned systems work with untyped relational phrases. The PPDB paraphrase database [11] uses a very different approach, employing bilingual texts. However, they concentrate on paraphrases of textual phrases rather than finding synonymous or similar relations.

The above approaches are capable of performing semantic clustering at large scale. However, they are limited in terms of the relational features used. OntExt [3] uses richer contextual information for clustering but can extract and cluster relations of only one pair of types at one time. There is a need for more powerful methods which can incorporate many types of relational features, yet can solve large-scale semantic clustering problems.

To address this challenge, in this paper we present a method for clustering relational phrases using a statistical relational learning system called Probabilistic Soft Logic (PSL) [1]. The proposed method has several advantages. First, the PSL modeling language allows us to easily build a rich model incorporating both similarity measures and relational features. Moreover, it is efficient enough to be applied to a dataset containing 200,000 relational phrases.

We perform a quantitative evaluation of the proposed PSL model on a small dataset. The performance of our approach is compared against a set of baselines, including textual similarity and argument overlap, demonstrating the efficacy of the technique. Additionally, we report the outcome of the PSL method on the large-scale dataset extracted by PATTY [4], illustrating that the proposed method is highly scalable.

2 Semantic Clustering using Probabilistic Soft Logic

The proposed models begin with the ideas of [4] and [5], and build upon them using relational learning techniques. This is accomplished using probabilistic soft logic, a declarative language for specifying templates for probabilistic graphical models. The resulting models, known as hinge-loss Markov random fields (HL-MRFs) [12], define probability densities over continuous random variables in the range [0,1]. Inference in this setting is a convex optimization task, which can be solved efficiently and at scale. PSL has been successfully applied to problems such as entity resolution [13], trust in social networks [14], ontology alignment [15], and social group modeling [16]. A PSL model is specified by a set of weighted first order logical rules. The resulting Markov random field gives higher probability density to states where the rules are closer to being satisfied, as measured by a continuous relaxation of Boolean logic. We now describe our proposed approach.

Predicates: The PSL models introduced in this work use the following predicates:

- $args(R_1, X, Y)$ - is true when relational phrase R_1 occurs with domain argument X and range argument Y.
- $types(R_1, T_1, T_2)$ - is true when T_1 and T_2 are domain and range types of relation R_1.
- $phrase(R_1, STR_1)$ - is true when STR_1 is a textual representation of relational phrase R_1. It is an implementation detail which allows us to separate numerical identifier R_1 from string STR_1 in the logic of our models.
- $simPattern(STR_1, STR_2)$ - is true when Jaccard similarity of tokens of strings STR_1 and STR_2 is above a threshold.
- $similar(R_1, R_2)$ - an open predicate which is true when relational phrases R_1 and R_2 belong to the same cluster.

PSL rules: The rules used in the PSL models are shown in Table 1. We employed standard similarity functions for argument similarity and textual similarity (rules 1,2,3,4). Rule 5 ensures that the similarity relation between relational phrases is transitive. Additionally, we define a prior on the inference predicate $similar$ (rule 6). It says that we should assume that two phrases are not $similar$ with a small weight. This can be overridden with evidence as defined in the other rules.

PSL models: We have developed three sets of rules used by the PSL framework The full list of rules used by our models is shown in Table 1. Each model runs in two stages – weight learning and inference. Weight learning is performed on a separate training data set. In the inference stage,

Id	Name	Rule
1	Argument similarity (without types)	$args(R_1, X, Y) \wedge args(R_2, X, Y) \Rightarrow similar(R_1, R_2)$
2	Textual similarity (without types)	$phrase(R_1, STR_1) \wedge phrase(R_2, STR_2) \wedge simPattern(STR_1, STR_2) \Rightarrow similar(R_1, R_2)$
3	Argument similarity and type compatibility	$args(R_1, X, Y) \wedge args(R_2, X, Y) \wedge types(R_1, T_1, T_2) \wedge types(R_2, T_1, T_2) \Rightarrow similar(R_1, R_2)$
4	Textual similarity and type compatibility	$phrase(R_1, STR_1) \wedge phrase(R_2, STR_2) \wedge simPattern(STR_1, STR_2) \wedge types(R_1, T_1, T_2) \wedge types(R_2, T_1, T_2) \Rightarrow similar(R_1, R_2)$
5	Transitivity rule	$similar(R_1, R_2) \wedge similar(R_2, R_3) \Rightarrow similar(R_1, R_3)$
6	Negative prior	$\neg similar(R_1, R_2)$

Table 1: Rules used in PSL models

for each model, the defined rules with learned weights are applied to the test data. We used the following models:

- `PSL_sim`: In this model we use argument and textual similarity without types (rules 1,2,6)
- `PSL_types`: In this model we add to `PSL_sim` information about the semantic types of the domain and the range (rules 3,4,6)
- `PSL_trans`: This model extends `PSL_types` with transitivity rules (rules 3,4,5,6)

3 Evaluation

To assess the quality of PSL models we evaluated the proposed models and compared the results against baseline algorithms.

Baselines: To be able to compare PSL models we have implemented a set of baselines. All of these methods predict similarity between relational phrases. We consider four baselines:

- `Random`: for every pair of relations we randomly decide whether they are similar or not.
- `Arguments overlap`: two relations are similar if they occur with the same arguments. The higher the percentage of common arguments is, the more similar the relations are. This follows the distributional hypothesis [9]. For example, if the relations "is married to" and "is husband of" occurred only with arguments "George Clooney" and "Amal Alamuddin" then they would be clustered together because of 100% arguments overlap. The threshold above which two relations are clustered together is set using training data with the grid search algorithm.
- `String similarity`: to determine the similarity of two relations we use the Jaccard similarity of the sets of tokens of their textual representations. Again, two relations are said to be similar if their textual similarity is above a threshold, chosen using a grid search.
- `String & types similarity`: we use string similarity and add an additional feature, types compatibility. Type compatibility introduces a constraint which requires that two relations can be similar only if the semantic types of the domains of each relation are equal and the semantic types of the ranges are also equal.

Dataset: We prepared the ground truth semi-automatically based on the clusters of relational phrases produced by the PATTY system [4]. For the first experiment we used all together 526 relational phrases which were divided into training and test sets. The training set contains 399 phrases and 3,454 argument-relation-argument triples. These phrases were divided into 229 clusters. The test set contains 127 phrases and 1,494 argument-relation-argument triples. These phrases were divided into 58 clusters.

Evaluation setup: We treat our problem in two ways – first as a link prediction task and second as a clustering problem. In the first setting the goal is to predict which relational phrases are similar. For every pair of relational phrases the algorithms determine whether there should be a similarity link between them. The similarity links form a similarity graph. Next we compare the produced similarity links against the similarity links in the ground truth. As the evaluation metrics for the link prediction task we use F1 score and area under the receiver operating characteristic curve (AUC).

In the second setting the goal is to form clusters of synonymous relational phrases. In this case we take the output of the previous setting and organize phrases into clusters by applying a connected component detection algorithm [17]. Relational phrases which are in the same component are put to the same cluster. For the purpose of the evaluation we assume that phrases inside a cluster are all connected with similarity links. Again, we compare similarity links against the ground truth and compute F1 score and AUC. Additionally, we compute Normalized Mutual Information (NMI) [18] – a metric used specifically for the clustering problem.

In order to compute statistical significance the experiment was repeated 20 times on random subsets of the patterns in the test data. We performed a paired t-test for all metrics (F1, AUC for link prediction and clustering; NMI for clustering) of the PSL models results against baselines and obtained p-values below 0.05.

Results: The results of the evaluation are shown in Table 2. The `String & types similarity` method performs the best out of all baselines in terms of all metrics and settings. The `PSL_types` and `PSL_trans` models have higher scores than baselines in both link prediction and clustering tasks. Moreover, in the link prediction task, we can see the difference between `PSL_types` and `PSL_trans` models. This shows us the influence of the `transitivity rule`.

After the application of the connected components detection algorithm there is no difference between `PSL_types` and `PSL_trans` models. This means that the transitivity rule in PSL models tends to play the same function as the connected components detection algorithm. However, PSL allows us to incorporate it in a single model rather than in a few separate algorithms. The cause of the equal scores of `PSL_types` and `PSL_trans` could be the size of the dataset. We do not observe big clusters in the data, therefore the transitivity cannot be often applied.

	Link prediction		Clustering		
	F1	AUC	F1	AUC	NMI
Random	0.0174	0.5000	0.0221	0.5000	0.8020
Arguments overlap	0.1204	0.5337	0.1167	0.5349	0.8740
String similarity	0.4017	0.6281	0.5878	0.7312	0.9236
String & types similarity	0.4085	0.6283	0.6350	0.7326	0.9321
PSL_sim	0.3675	0.6146	0.5684	0.7205	0.9193
PSL_type	0.6690	0.7538	0.7151	0.8011	0.9448
PSL_trans	0.7290	0.8017	0.7151	0.8011	0.9448

Table 2: Evaluation

Studies on the PATTY dataset: We also investigated the performance of the best performing PSL model on a larger amount of data. To increase the speed of the algorithm we used precomputed similarities of textual representations of relational phrases. We used a subset of the patterns taken using PATTY by Nakashole et al. [4]. The subset consisted of 200,000 patterns, contained information about 1,158,417 argument-relation-argument triples and was originally organized into 162,289 clusters. As a result of running the algorithm we obtained 144,634 clusters. Some example clusters created during this process are shown in Table 3.

Cluster	Domain	Phrases	Range
1	sovereign	became emperor as; ascended the throne as; succeeded as; took the throne as	head of state
2	person	be consecrated by; enthroned as; also consecrated;	priest
3	actor	had starred in; best known for playing on; again starred in;	event
4	person	and defeated the; successfully defended against;	team

Table 3: Example clusters of phrases

4 Conclusion and Future Work

In this paper, we demonstrate an approach for semantic clustering of relational phrases. This approach uses the PSL framework for modeling similarities between phrases. In the experiments, we showed that our method outperforms several baselines and is capable of reconstructing a clustering

performed by PATTY [4]. Moreover, we applied PSL to a dataset whose size is comparable with datasets used by state of the art systems for extraction and clustering of relational phrases. Since we used basic features and basic similarity measures there is still the potential for further improvement. As future work, we can use the expressiveness of PSL to incorporate other similarity measures into our models. Furthermore, other sources of relational phrases can be used (e.g. WiseNet [5]). Finally, we also plan to perform a human evaluation of the results of running PSL on the PATTY dataset.

Acknowledgments: This work was partially supported by National Science Foundation (NSF) grant IIS1218488 and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC00337. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, IARPA, DoI/NBC, or the U.S. Government.

References

[1] Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*, 2012.

[2] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[3] Thahir Mohamed, Estevam R. Hruschka Jr., and Tom M. Mitchell. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1447–1455, 2011.

[4] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1135–1145, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[5] Andrea Moro and Roberto Navigli. Wisenet: Building a wikipedia-based semantic network with ontologized relations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1672–1676, New York, NY, USA, 2012. ACM.

[6] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL 2013*, pages 74–84, 2013.

[7] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[8] Dekang Lin and Patrick Pantel. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 323–328, New York, NY, USA, 2001. ACM.

[9] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[10] Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *J. Artif. Int. Res.*, 34(1):255–296, March 2009.

[11] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[12] Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Uncertainty in Artificial Intelligence (UAI)*, 2013.

[13] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *International Semantic Web Conference (ISWC)*, 2013.

[14] Bert Huang, Angelika Kimmig, Lise Getoor, and Jennifer Golbeck. Probabilistic soft logic for trust analysis in social networks. In *International Workshop on Statistical Relational Artificial Intelligence (StaRAI 2012)*, 2012.

[15] Matthias Broecheler and Lise Getoor. Probabilistic similarity logic. In *International Workshop on Statistical Relational Learning*, 2009.

[16] Bert Huang, Stephen H. Bach, Eric Norris, Jay Pujara, and Lise Getoor. Social group modeling with probabilistic soft logic. In *NIPS Workshop on Social Network and Social Media Analysis: Methods, Models, and Applications*, 2012.

[17] John E. Hopcroft and Robert Endre Tarjan. Efficient algorithms for graph manipulation [H] (algorithm 447). *Commun. ACM*, 16(6):372–378, 1973.

[18] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, chapter 16.3 Evaluation of clustering. Cambridge University Press, 2008.