# Using Graphs of Classifiers to Impose Constraints on Semi-supervised Relation Extraction

**Lidong Bing** and **William W. Cohen** and **Bhuwan Dhingra**
School of Computer Science
Carnegie Mellon Univeristy
{lbing, wcohen, bdhingra}@cs.cmu.edu

**Richard C. Wang**
US Development Center
Baidu USA
richardwang@baidu.com

## Abstract

We propose a general approach to modeling semi-supervised learning constraints on unlabeled data. Both traditional supervised classification tasks and many natural semi-supervised learning heuristics can be approximated by specifying the desired outcome of walks through a graph of classifiers. We demonstrate the modeling capability of this approach in the task of relation extraction, and experimental results show that the modeled constraints achieve better performance as expected.

## 1 Introduction

Semi-supervised learning (SSL) methods often operate by introducing "soft constraints" on how a learned classifier will behave at points, or clusters of points, associated with unlabeled instances. For example, logistic regression with entropy regularization (Grandvalet and Bengio, 2004) and transductive SVMs (Joachims, 1999) constrain the classifier to make confident predictions at unlabeled points, and many graph-based SSL approaches require that the instances associated with the endpoints of an edge have similar labels (Zhu et al., 2003; Talukdar and Crammer, 2009). Other weakly-supervised methods also can be viewed as imposing constraints predictions made by a classifier: for instance, in distantly-supervised information extraction, constraints sometimes are imposed which requires that the classifier, when applied to the set $S$ of mentions of an entity pair that is a member of relation $r$, classify at least one mention in $S$ as a positive instance of $r$ (Hoffmann et al., 2011). Different constraints (and different assumptions about the loss function for the learner) lead to different SSL algorithms.

In this paper, we propose a general approach to modeling such constraints. In particular, we show that many types of constraints can be modeled by *specifying the desired behavior of random walks through a graph of classifiers*. In the graph, nodes correspond to relational conditions on small subsets of the data, and edges are annotated by feature vectors. Feature weights, combined with the feature vector at each edge and a non-linear postprocessing step, define a weighting of edges in the graph, and hence a transition function for a random walk. We will argue that traditional supervised classification tasks, as well as many natural SSL heuristics, can be approximated by specifying the desired outcome of walks through this graph.

Below we will make this notion precise. We will also define a succinct declarative language for specifying these models, and introduce a corresponding graphical "plate" language for the models. We then present results obtained by optimizing performance on an appropriate ensemble of graphs for the task of relation extraction.

## 2 Specifying SSL Tasks

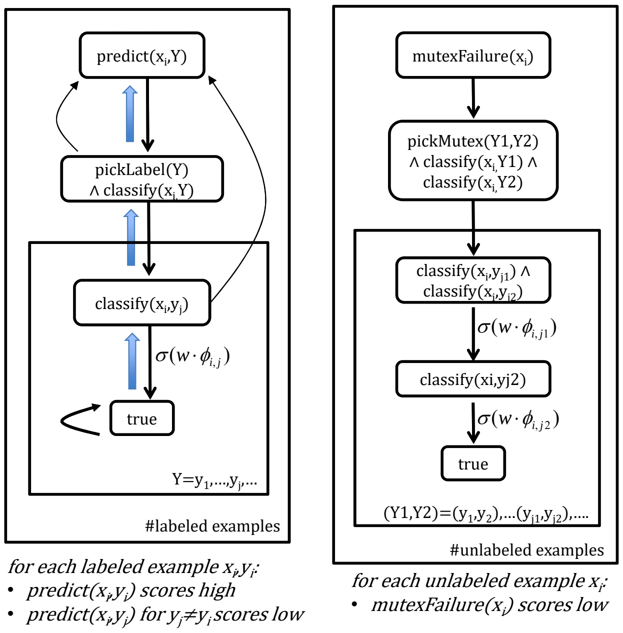
### 2.1 An Example: Intuition

We begin with an simple example. The left-hand side of Figure 1 illustrates how a traditional supervised classification can be expressed as programs in ProPPR (Wang et al., 2013), a probabilistic first-order language. In ProPPR, following the convention used on logic programming, capital letters are

```
predict(X,Y) ←
    pickLabel(Y) ∧
    classify(X,Y).
classify(X,Y) ← true
    { f(W,Y): hasFeature(X,W) }.
```

```
mutexFailure(X) ←
    pickMutex(Y1,Y2) ∧
    classify(X,Y1) ∧
    classify(X,Y2).
```

**Figure 1:** Declarative specifications of the models for supervised learning, on the left, and for a mutual-exclusivity constraint, on the right.



predict($x_i$,Y)
pickLabel(Y) ∧ classify($x_i$,Y)
classify($x_i$,$y_j$)
$\sigma(w \cdot \phi_{i,j})$
true
Y=$y_1$,...,$y_j$,...
#labeled examples
for each labeled example $x_i$,$y_i$:
• predict($x_i$,$y_i$) scores high
• predict($x_i$,$y_j$) for $y_j \neq y_i$ scores low
mutexFailure($x_i$)
pickMutex(Y1,Y2) ∧ classify($x_i$,Y1) ∧ classify($x_i$,Y2)
classify($x_i$,$y_{j1}$) ∧ classify($x_i$,$y_{j2}$)
$\sigma(w \cdot \phi_{i,j1})$
classify(xi,yj2)
$\sigma(w \cdot \phi_{i,j2})$
true
(Y1,Y2)=($y_1$,$y_2$),...($y_{j1}$,$y_{j2}$),....
#unlabeled examples
for each unlabeled example $x_i$:
• mutexFailure($x_i$) scores low

**Figure 2:** Plate diagrams for supervised learning, on the left, and a mutual-exclusivity constraint, on the right.

implicitly variables, and are universally quantified when they appear in the "head" of a rule, and rules can also be annotated with a set of *features*, which then weighted to define a strength for the rule. The symbol *true* is a goal that always succeeds, and we omit, for brevity, the problem specific definition of *pickLabel(Y)*, which would consist of rules for each possible label $y_i$ (relation types), i.e. *pickLabel*($y_1$) ← *true*, ..., *pickLabel*($y_K$) ← *true*.

ProPPR programs, like Prolog programs, are associated with a backward-chaining proof process, and like Prolog programs, can be read either as logical constraints, or as a non-deterministic program, which is invoked when a query is submitted. If the queries processed by the program are all of the form *predict*($x_i$,*Y*) where $x_i$ is a constant, the theory on the left-hand side of Figure 1 can be interpreted as saying: (1) To prove the goal of the form *predict*($x_i$,*Y*)—i.e., to predict a label $Y$ for the instance $x_i$—non-deterministically pick a possible class label $y_j$, and then prove the goal *classify*($x_i$,$y_i$); (2) proofs for every the goal *classify*($x_i$,$y_j$) immediately succeed, with a strength based on a weighted combination of the features in the set $\{f(w, y_j) : hasFeature(x_i, w)\}$. This set is encoded in the usual way as a sparse vector $\phi_{x_i,y_j}$, with one dimension for every object of the form $f(w, y_j)$ where $y_j$ is a class label and $w$ is a vocabulary word. For example, if the vocabulary contains the word *hope* and *sports* is a possible label, then one feature in $\phi_{x_i,y_j}$ might be active exactly when document $x_i$ contains the word $hope$ and $y_j$ = *sports*. The set of proofs associated with this theory are described by the plate diagram on the left-hand side of Figure 2. Although the nodes are *not* associated with logical variables, the repetition suggested by the plates has the same meaning as in graphical models, and the heavy blue upward-pointing arrows denote logical implication.

In ProPPR[1] it is possible to train weights to maximize or minimize the score of a particular query response: i.e., one can say that for the query *predict*($x_i, Y$) responses where $Y = y_{j*}$ are "positive" and responses where $Y = y_{j'}$ for $j' \neq j*$ are "negative". The training data needed for the supervised learning case is indicated in the bottom of the left-hand of the plate diagram. Learning is performed by stochastic gradient descent (Wang et al., 2013).

## 2.2 Unlabeled Data and Low-density Boundaries

We finally turn to the right-hand sides of Figures 1 and 2. These can be viewed as a sort of consistency test to be applied to an *unlabeled* example $x_i$. In ordinary classification tasks, any two distinct classes $y_j$ and $y_{j'}$ should be mutually exclusive. The theory on the right-hand side of Figure 1

[1]ProPPR's semantics are defined by a slightly different graph, which contains the same set of nodes as the proof graph, but is weighted, and has a different edge set—namely, the downward-pointing black arrows, which run opposite to the implication edges. For this example, these edges describe a forest, with one tree for each labeled example $x_i$. The forest is further augmented with a self-loop on each *true* node, and a "reset" edge that returns to the root for each non-*true* node. To simplify, the reset and self-loop edges are only shown in the first plate diagram.

asserts that a “mutual exclusion failure” (mutexFailure) occurs if $x_i$ can be classified into two distinct classes. (Again there is a problem specific definition of *pickMutex(Y1,Y2)*, which would consist of trivial rules for each possible distinct label pair $y_j$, and $y_{j'}$.) The corresponding plate diagram is shown in Figure 2. To (softly) enforce this constraint, we need only introduce negative examples for each unlabeled example $x_i$, specifying that proofs for the goal *mutexFailure($x_i$)* should have a low scores.

Conceptually, this constraint encodes a common bias of SSL systems, namely, that the decision boundaries should be drawn in low-probability regions of the space. In this case, if a decision boundary is close to an unlabeled example, then more than one *classify* goal with succeed with a high score.

## 2.3 Other SSL constraints

The framework describes above is flexible enough to handle many types of constraints. Here we are primarily interested in constraints associated with relation extraction. Before introducing these constraints, we first describe our task, relation extraction for entity-centric corpora.

Each document in an entity-centric corpus describes aspects of a particular entity (called *subject or title entity*), e.g. each Wikipedia article is such a document. Relation extraction from a entity-centric document is reduced to predicting the relation between the subject entity and an entity mention in the document. For example, for a drug article, if the target relation is sideEffects, we need to predict for each candidate (extracted from a single sentence) whether it is a side effect of this drug. If no such relation holds, we predict the special label “Other”. Besides the entropy regularization constraint, i.e., mutexFailure introduced above, there are several other constraints that could be helpful for this task.

**Sentence constraint**. For each sentence, we constrain that only one mention (here a mention can also refer to a coordinate-term list such as “vomiting, headache and nausea”) should be labeled as a particular relation:

sentFailure(X1,X2) $\leftarrow$ pickRealLabel(Y1)$\wedge$
pickRealLabel(Y2)$\wedge$classify(X1,Y1)$\wedge$classify(X2,Y2),

where pickRealLabel(Y) picks a label other than “Other”, X1 and X2 are a pair of mentions extracted from a single sentence. This constraint penalizes extracting multiple relation objects from a single sentence.

**Document constraint.** If an entity string appears as multiple mentions in one document, they should have the same relation label (relative to the subject entity), or some of them have “Other” label:

docFailure(X1,X2) $\leftarrow$ pickMutex(Y1,Y2)$\wedge$
pickRealLabel(Y1)$\wedge$pickRealLabel(Y2)$\wedge$
classify(X1,Y1)$\wedge$classify(X2,Y2).

**Section title constraint.** In some entity-centric corpora, the content of a document is organized into different sections. This constraint basically says if two mentions appear in the same section (currently determined simply by matching section titles) of two documents, they should have the same relation label, relative to their own document subjects:

titleFailure(X1,X2) $\leftarrow$ pickMutex(Y1,Y2)$\wedge$
pickRealLabel(Y1)$\wedge$pickRealLabel(Y2)$\wedge$
classify(X1,Y1)$\wedge$classify(X2,Y2).

## 3 Experiments

### 3.1 Settings

**Corpora**. Our drug corpus, DailyMed, is downloaded from dailymed.nlm.nih.gov which contains 28,590 XML documents, each of which describes a drug that can be legally prescribed in the United States. Our disease corpus, WikiDisease, is extracted from a Wikipedia dump of May 2015 and it contains 8,596 disease articles. We extract usedToTreat, conditionsThisMayPrevent, and sideEffects relations for the drug domain; treatments, symptoms, riskFactors, causes, and preventionFactors relations for the disease domain.

**Preprocessing and features**. We use the GDep parser (Sagae and Tsujii, 2007), a dependency parser trained on the GENIA Treebank, to parse the corpora. We use a simple POS-based chunker to extract NP mentions, and also extract a list for each coordinating conjunction that modifies a nominal (a list is regarded as a compound mention). We use the same feature generator for both mentions and lists. Shallow features include: tokens in the NPs, and character prefixes/suffixes of these tokens; tokens from the sentence containing the NP; and tokens and bigrams from a window around the NPs. From the dependency parsing, we also find the verb which is the closest ancestor of the head of the NP, all modifiers of this verb, and the path to this verb. For a list, the dependency features are computed relative to the head of the list.

**Evaluation dataset**. We manually labeled 10 pages from WikiDisease and 10 pages from DailyMed. The annotated text fragments are those NPs that are the second argument values of those 8 relations, with the title drug or disease entity of the corresponding document as the relation subject. In total, there are 436 triple facts for the disease domain and 320 triple facts for the drug domain. A pipeline's task is to extract values of the second arguments of relations from a given document.

## 3.2 Training Data with Distant Supervision

We extract triples from Freebase as supervision to distantly label training examples. If the subject of a triple matches with a drug or disease title entity in a corpus and its object value also appears in that document, it is extracted. In total, we get 2022, 2453, 905, 753, and 164 triples for 5 disease relations respectively, and 3112, 315, and 265 triples for 3 drug relations, respectively.

Each triple is used to label the document whose subject entity is the same as the triple subject. For instance, triple sideEffects(Aspirin,heartburn) will label a mention "heartburn" from the Aspirin article as an example of sideEffects relation. This raw data is very noisy (Bing et al., 2015; Bing et al., 2016), so we add a distillation step. We first distantly label these relations in two small structured corpora, namely, WebMD for drug and MayoClinic for disease.[2] They have well-defined section information, which can be matched with target relations. We only label usedToTreat and conditionsThisMayPrevent from the "Uses" section, and label sideEffects from "Side Effects" section of WebMD. Similarly, the disease relations are labeled from "Treatments and drugs","Symptoms", "Risk factors", "Causes", and "Prevention" sections of MayoClinic. After that we build a graph containing examples from both corpora of a domain, and do label propagation in this graph with the section-labeled examples as seeds. We take top 2,000 examples from each WikiDisease relation and top 800 examples from each DailyMed relation as training data. We randomly pick 2,000 and 800 examples that are not distantly labeled by any relation as "Other" examples.

[2]WebMD is collected from www.webmd.com, and each drug page has the same seven sections, such as Uses, Side Effects, Precautions, etc. WebMD contains 2,096 pages. MayoClinic is collected from www.mayoclinic.org. The sections of MayoClinic pages include Symptoms, Causes, Risk Factors, Treatments and Drugs, Prevention, etc. MayoClinic contains 1,117 pages.

## 3.3 Algorithms Compared

Our SSL framework allows many constraints to be formulated, leading to several SSL methods: SSL_m uses only the mutex constraint; SSL_s, only the sentence constraint; SSL_d, only the document constraint; SSL_t, only the section title constraint. We also consider some combinations of these: SSL_sd; SSL_st; SSL_dt; and SSL_sdt. All the SSL pipelines employ the evaluation pages as unlabeled data for those constraints (i.e., they are used transductively).

As one baseline, we compare to a standard supervised learning pipeline, SL, which learns a classifier with no constraints using ProPPR. We also compare against three existing methods: *MultiR*, (Hoffmann et al., 2011) which models each relation mention separately and aggregates their labels using a deterministic OR; *Mintz++* from (Surdeanu et al., 2012), which improves on the original model from (Mintz et al., 2009) by training multiple classifiers, and allowing multiple labels per entity pair; and *MIML-RE* (Surdeanu et al., 2012) which has a similar structure to *MultiR*, but uses a classifier to aggregate the mention level predictions into an entity pair prediction. We used the publicly available code from the authors [3] for the experiments. Since these methods do not distinguish between structured and unstructured corpora, we used the union of these corpora in our experiments. We found that the performance of these methods varies significantly with the number of negative examples used during training, and hence we tuned these and other parameters, including the number of epochs (for both *MultiR* and *MIML-RE*) and the number of training folds for *MIML-RE*, directly on the evaluation data and report their best performance.

Finally we compare with our previous system, DIEBOLDS, (Bing et al., 2016) which uses docu-

[3]http://aiweb.cs.washington.edu/ai/raphaelh/mr/ and http://nlp.stanford.edu/software/mimlre.shtml

ment structure to construct a different label propagation graph. Briefly, DIEBOLDS first builds a bipartite graph from the merged structured corpus and target corpus, and then performs relation type propagation with the extracted lists (including singleton ones) and their items. One set of vertices correspond to relation mentions. The other set of vertexes are identifiers for the lists. Additional couplings use the document structure and BOW context features of pairs. Label propagation uses the subject-NP pairs distantly labeled with relation seeds as starting points, and then binary classifiers are trained with the top N mentioned as score by label propagation.

### 3.4 Results

The SL and SSL pipelines can classify both singleton and coordinate lists. After that, lists are broken into items, i.e. NPs, for evaluation. We evaluate the performance of different pipelines from IR perspective, with a title entity (i.e., document name) and a relation together as a query, and extracted NPs as retrieval results. The predicted probability by ProPPR serves as the ranking score inside each query. The results evaluated by precision, recall and F1 measure are given in Table 1. DIEBOLDS' results are extracted from (Bing et al., 2016), since the evaluation data is the same.

Among the individual constraints, SSL_s, SSL_d, and SSL_t are found to improve the performance over SL (which is a strong baseline, perhaps because of careful use of structured documents in our distant labeling procedure.) These SSL approaches lead to higher precision, showing that adding these constraints does reduce false positives. The sentence constraint is the most helpful for better precision: mentions from the same sentence share token features from sentence content, which often misleads the classifiers, and the sentence constraint is designed to penalize such cases. SSL_m is useful for improving recall, but its precision is much lower than SL. Unlike the other constraints, we note that SSL_m is a domain-independent heuristic: it simply encourages confident decision on unlabeled examples. This is a useful heuristic in many cases, and in particular encourages classifiers that lie in low-density areas of the example space; we conjectiure that this is inapproproate for this task because the data is noisy and the classes are not well-separated.

| | Disease | | | Drug | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| DIEBOLDS | 0.143 | 0.372 | 0.209 | 0.050 | **0.435** | 0.090 |
| MultiR | 0.198 | 0.333 | 0.249 | 0.156 | 0.138 | 0.146 |
| Mintz++ | 0.192 | 0.353 | 0.249 | 0.177 | 0.178 | 0.178 |
| MIML-RE | 0.211 | 0.360 | 0.266 | 0.167 | 0.160 | 0.163 |
| SL | 0.247 | 0.353 | 0.290 | 0.288 | 0.368 | 0.323 |
| SSL_m | 0.191 | **0.382** | 0.255 | 0.207 | 0.418 | 0.277 |
| SSL_s | 0.284 | 0.317 | 0.299 | 0.294 | 0.367 | 0.326 |
| SSL_d | 0.257 | 0.350 | 0.296 | 0.293 | 0.366 | 0.325 |
| SSL_t | 0.257 | 0.362 | 0.301 | 0.292 | 0.364 | 0.324 |
| SSL_sd | **0.294** | 0.318 | 0.306 | 0.291 | 0.367 | 0.325 |
| SSL_st | 0.289 | 0.332 | 0.309 | 0.300 | 0.376 | 0.334 |
| SSL_dt | 0.264 | 0.369 | 0.308 | 0.299 | 0.384 | 0.336 |
| SSL_sdt | 0.292 | 0.335 | **0.312** | **0.304** | 0.378 | **0.337** |

**Table 1:** Average results from 3 runs.

Combining the individual constraints can further improve the results. SSL_sdt is the most effective pipeline. Compared with SL, it achieves 4.3% and 7% relative improvements for drug domain and disease domain, respectively. Compared with MultiR, Mintz++, and MIML-RE, the relative improvements are about 17% to 25% on the disease domain, and 89% to 131% on the drug domain. DIEBOLDS achieves the highest recall values, however, its precision is much lower.

## 4 Conclusions

We proposed a general approach to modeling SSL constraints. It can approximate traditional supervised learning and many natural SSL heuristics by specifying the desired outcome of walks through a graph of classifiers. An application case of this approach is given by modeling the task of relation extraction. There are a few open questions to explore: adding hyperparameters (e.g., different weights for different constraints); adding more control over the supervised loss versus the constraint-based loss; and testing the approach on more tasks.

## Acknowledgments

This work was funded by grants from Baidu USA and Google.

## References

Lidong Bing, Sneha Chaudhari, Richard Wang C, and William W Cohen. 2015. Improving distant supervision for information extraction using label propagation through lists. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 524–529, Lisbon, Portugal, September. Association for Computational Linguistics.

Lidong Bing, Mingyang Ling, Richard C. Wang, and William W. Cohen. 2016. Distant IE by bootstrapping using lists and document structure. In *The Thirtieth AAAI Conference on Artificial Intelligence*, AAAI '16.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2129–2138. ACM.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of ICML-03, the 20th International Conference on Machine Learning*.