A Linear Programming Formulation for Global Inference in Natural Language Tasks

Dan Roth **Wen-tau Yih**
Department of Computer Science
University of Illinois at Urbana-Champaign
{danr, yih}@uiuc.edu

Abstract

Given a collection of discrete random variables representing outcomes of learned local predictors in natural language, e.g., named entities and relations, we seek an optimal global assignment to the variables in the presence of general (non-sequential) constraints. Examples of these constraints include the type of arguments a relation can take, and the mutual activity of different relations, etc. We develop a linear programming formulation for this problem and evaluate it in the context of simultaneously learning named entities and relations. Our approach allows us to efficiently incorporate domain and task specific constraints at decision time, resulting in significant improvements in the accuracy and the "human-like" quality of the inferences.

1 Introduction

Natural language decisions often depend on the outcomes of several different but mutually dependent predictions. These predictions must respect some constraints that could arise from the nature of the data or from domain or task specific conditions. For example, in part-of-speech tagging, a sentence must have at least one verb, and cannot have three consecutive verbs. These facts can be used as constraints. In named entity recognition, "no entities can overlap" is a common constraint used in various works (Tjong Kim Sang and De Meulder, 2003).

Efficient solutions to problems of these sort have been given when the constraints on the predictors are *sequential* (Dietterich, 2002). These solutions can be categorized into the following two frameworks. *Learning global models* trains a probabilistic model under the constraints imposed by the domain. Examples include variations of HMMs, conditional models and sequential variations of Markov random fields (Lafferty et al., 2001). The other framework, *inference with classifiers* (Roth, 2002), views maintaining constraints and learning classifiers as separate processes. Various local classifiers are trained without the knowledge of constraints. The predictions are taken as input on the inference procedure which then finds the best global prediction. In addition to the conceptual simplicity of this approach, it also seems to perform better experimentally (Tjong Kim Sang and De Meulder, 2003).

Typically, efficient inference procedures in both frameworks rely on dynamic programming (e.g., Viterbi), which works well in sequential data. However, in many important problems, the structure is more general, resulting in computationally intractable inference. Problems of these sorts have been studied in computer vision, where inference is generally performed over low level measurements rather than over higher level predictors (Levin et al., 2002; Boykov et al., 2001).

This work develops a novel *inference with classifiers* approach. Rather than being restricted on sequential data, we study a fairly general setting. The problem is defined in terms of a collection of discrete random variables representing binary relations and their arguments; we seek an optimal assignment to the variables in the presence of the constraints on the binary relations between variables and the relation types.

The key insight to this solution comes from recent techniques developed for approximation algorithms (Chekuri et al., 2001). Following this work, we model inference as an optimization problem, and show how to cast it as a linear program. Using existing numerical packages, which are able to solve very large linear programming problems in a very short time[1], inference can be done very quickly.

Our approach could be contrasted with other ap-

[1]For example, (CPLEX, 2003) is able to solve a linear programming problem of 13 million variables within 5 minutes.

proaches to sequential inference or to general Markov random field approaches (Lafferty et al., 2001; Taskar et al., 2002). The key difference is that in these approaches, the model is learned globally, under the constraints imposed by the domain. In our approach, predictors do not need to be learned in the context of the decision tasks, but rather can be learned in other contexts, or incorporated as background knowledge. This way, our approach allows the incorporation of constraints into decisions in a dynamic fashion and can therefore support task specific inferences. The significance of this is clearly shown in our experimental results.

We develop our models in the context of natural language inferences and evaluate it here on the problem of *simultaneously* recognizing named entities and relations between them.

1.1 Entity and Relation Recognition

This is the problem of recognizing the *kill (KFJ, Oswald)* relation in the sentence "`J. V. Oswald was murdered at JFK after his assassin, R. U. KFJ...`" This task requires making several local decisions, such as identifying named entities in the sentence, in order to support the relation identification. For example, it may be useful to identify that Oswald and KFJ are *people*, and JFK is a *location*. This, in turn, may help to identify that the *kill* action is described in the sentence. At the same time, the relation *kill* constrains its arguments to be *people* (or at least, not to be *location*s) and helps to enforce that Oswald and KFJ are likely to be *people*, while JFK is not.

In our model, we first learn a collection of "local" predictors, e.g., entity and relation identifiers. At decision time, given a sentence, we produce a global decision that optimizes over the suggestions of the classifiers that are active in the sentence, known constraints among them and, potentially, domain or tasks specific constraints relevant to the current decision.

Although a brute-force algorithm may seem feasible for short sentences, as the number of entity variable grows, the computation becomes intractable very quickly. Given n entities in a sentence, there are $O(n^2)$ possible relations between them. Assume that each variable (entity or relation) can take l labels ("none" is one of these labels). Thus, there are l^{n^2} possible assignments, which is too large even for a small n.

When evaluated on simultaneous learning of named entities and relations, our approach not only provides a significant improvement in the predictors' accuracy; more importantly, it provides *coherent* solutions. While many statistical methods make "stupid" mistakes (i.e., inconsistency among predictions), that no human ever makes, as we show, our approach improves also the *quality* of the inference significantly.

The rest of the paper is organized as follows. Section 2 formally defines our problem and section 3 describes the computational approach we propose. Experimental results are given in section 4, followed by some discussion and conclusion in section 5.

2 The Relational Inference Problem

We consider the relational inference problem within the *reasoning with classifiers* paradigm, and study a specific but fairly general instantiation of this problem, motivated by the problem of recognizing named entities (e.g., persons, locations, organization names) and relations between them (e.g. work_for, located_in, live_in). We consider a set $\mathcal{V}$ which consists of two types of variables $\mathcal{V} = \mathcal{E} \cup \mathcal{R}$. The first set of variables $\mathcal{E} = \{E_1, E_2, \cdots, E_n\}$ ranges $\mathcal{L}_\mathcal{E}$. The value (called "label") assigned to $E_i \in \mathcal{E}$ is denoted $f_{E_i} \in \mathcal{L}_\mathcal{E}$. The second set of variables $\mathcal{R} = \{R_{ij}\}_{\{1 \leq i,j \leq n; i \neq j\}}$ is viewed as binary relations over $\mathcal{E}$. Specifically, for each pair of entities E_i and E_j, $i \neq j$, we use R_{ij} and R_{ji} to denote the (binary) relations (E_i, E_j) and (E_j, E_i) respectively. The set of labels of relations is $\mathcal{L}_\mathcal{R}$ and the label assigned to relation $R_{ij} \in \mathcal{R}$ is $f_{R_{ij}} \in \mathcal{L}_\mathcal{R}$.

Apparently, there exists some constraints on the labels of corresponding relation and entity variables. For instance, if the relation is *live_in*, then the first entity should be a *person*, and the second entity should be a *location*. The correspondence between the relation and entity variables can be represented by a bipartite graph. Each relation variable R_{ij} is connected to its first entity E_i , and second entity E_j. We use $\mathcal{N}^1$ and $\mathcal{N}^2$ to denote the entity variables of a relation R_{ij}. Specifically, $E_i = \mathcal{N}^1(R_{ij})$ and $E_j = \mathcal{N}^2(R_{ij})$.

In addition, we define a set of constraints on the outcomes of the variables in $\mathcal{V}$. $\mathcal{C}^1 : \mathcal{L}_\mathcal{E} \times \mathcal{L}_\mathcal{R} \rightarrow \{0, 1\}$ constraint values of the first argument of a relation. $\mathcal{C}^2$ is defined similarly and constrains the second argument a relation can take. For example, (*born_in*, *person*) is in $\mathcal{C}^1$ but not in $\mathcal{C}^2$ because the first entity of relation *born_in* has to be a *person* and the second entity can only be a *location* instead of a *person*. Note that while we define the constraints here as Boolean, our formalisms in fact allows for stochastic constraints. Also note that we can define a large number of constraints, such as $\mathcal{C}^R : \mathcal{L}_\mathcal{R} \times \mathcal{L}_\mathcal{R} \rightarrow \{0, 1\}$ which constrain types of relations, etc. In fact, as will be clear in Sec. 3 the language for defining constraints is very rich – linear (in)equalities over $\mathcal{V}$.

We exemplify the framework using the problem of simultaneous recognition of named entities and relations in sentences. Briefly speaking, we assume a learning mechanism that can recognize entity phrases in sentences, based on local contextual features. Similarly, we assume a learning mechanism that can recognize the semantic relation between two given phrases in a sentence.

We seek an inference algorithm that can produce a coherent labeling of entities and relations in a given sentence. Furthermore, it follows, as best as possible the recommendation of the entity and relation classifiers, but also satisfies natural constraints that exist on whether specific entities can be the argument of specific relations, whether two relations can occur together at the same time, or any other information that might be available at the inference time (e.g., suppose it is known that entities A and B represent the same location; one may like to incorporate an additional constraint that prevents an inference of the type: "C lives in A; C does not live in B").

We note that a large number of problems can be modeled this way. Examples include problems such as chunking sentences (Punyakanok and Roth, 2001), coreference resolution and sequencing problems in computational biology. In fact, each of the components of our problem here, the separate task of recognizing named entities in sentences and the task of recognizing semantic relations between phrases, can be modeled this way. However, our goal is specifically to consider interacting problems at different levels, resulting in more complex constraints among them, and exhibit the power of our method.

The most direct way to formalize our inference problem is via the formalism of Markov Random Field (MRF) theory (Li, 2001). Rather than doing that, for computational reasons, we first use a fairly standard transformation of MRF to a discrete optimization problem (see (Kleinberg and Tardos, 1999) for details). Specifically, under weak assumptions we can view the inference problem as the following optimization problem, which aims to minimize the objective function that is the sum of the following two cost functions.

Assignment cost: the cost of deviating from the assignment of the variables $\mathcal{V}$ given by the classifiers. The specific cost function we use is defined as follows: Let l be the label assigned to variable $u \in \mathcal{V}$. If the marginal probability estimation is $p = P(f_u = l)$, then the assignment cost $c_u(l)$ is $-\log p$.

Constraint cost: the cost imposed by breaking constraints between neighboring nodes. The specific cost function we use is defined as follows: Consider two entity nodes E_i, E_j and its corresponding relation node R_{ij}; that is, $E_i = \mathcal{N}^1(R_{ij})$ and $E_j = \mathcal{N}^2(R_{ij})$. The constraint cost indicates whether the labels are consistent with the constraints. In particular, we use: $d^1(f_{E_i}, f_{R_{ij}})$ is 0 if $(f_{R_{ij}}, f_{E_i}) \in \mathcal{C}^1$; otherwise, $d^1(f_{E_i}, f_{R_{ij}})$ is ∞ [2]. Similarly, we use d^2 to force the consistency of the second argument of a relation.

[2] In practice, we use a very large number (e.g., 9^{15}).

Since we are seeking the most probable global assignment that satisfies the constraints, therefore, the overall cost function we optimize, for a global labeling f of all variables is:

$$\begin{aligned} C(f) &= \sum_{u \in \mathcal{V}} c_u(f_u) \\ &+ \sum_{R_{ij} \in \mathcal{R}} \left[d^1(f_{R_{ij}}, f_{E_i}) + d^2(f_{R_{ij}}, f_{E_j}) \right] \end{aligned} \quad (1)$$

3 A Computational Approach to Relational Inference

Unfortunately, it is not hard to see that the combinatorial problem (Eq. 1) is computationally intractable even when placing assumptions on the cost function (Kleinberg and Tardos, 1999). The computational approach we adopt is to develop a *linear programming* (LP) formulation of the problem, and then solve the corresponding *integer linear programming* (ILP) problem. Our LP formulation is based on the method proposed by (Chekuri et al., 2001). Since the objective function (Eq. 1) is not a linear function in terms of the labels, we introduce new binary variables to represent different possible assignments to each original variable; we then represent the objective function as a linear function of these binary variables.

Let $x_{\{u,i\}}$ be a $\{0,1\}$-variable, defined to be 1 if and only if variable u is labeled i, where $u \in \mathcal{E}, i \in \mathcal{L}_{\mathcal{E}}$ or $u \in \mathcal{R}, i \in \mathcal{L}_{\mathcal{R}}$. For example, $x_{\{E_1,2\}} = 1$ when the label of entity E_1 is 2; $x_{\{R_{23},3\}} = 0$ when the label of relation R_{23} is not 3. Let $x_{\{R_{ij},r,E_i,e_1\}}$ be a $\{0,1\}$-variable indicating whether relation R_{ij} is assigned label r and its first argument, E_i, is assigned label e_1. For instance, $x_{\{R_{12},1,E_1,2\}} = 1$ means the label of relation R_{12} is 1 *and* the label of its first argument, E_1, is 2. Similarly, $x_{\{R_{ij},r,E_j,e_2\}} = 1$ indicates that R_{ij} is assigned label r and its second argument, E_j, is assigned label e_2. With these definitions, the optimization problem can be represented as the following ILP problem (Figure 1).

Equations (2) and (3) require that each entity or relation variable can only be assigned one label. Equations (4) and (5) assure that the assignment to each entity or relation variable is consistent with the assignment to its neighboring variables. (6), (7), and (8) are the integral constraints on these binary variables.

There are several advantages of representing the problem in an LP formulation. First of all, linear (in)equalities are fairly general and are able to represent many types of constraints (e.g., the decision time constraint in the experiment in Sec. 4). More importantly, an ILP problem at this scale can be solved very quickly using current commercial LP/ILP packages, like (Xpress-MP, 2003) or (CPLEX, 2003). We introduce the general strategies of solving an ILP problem here.

$$\begin{aligned}\min \quad & \sum_{E\in\mathcal{E}}\sum_{e\in\mathcal{L}_{\mathcal{E}}} c_E(e)\cdot x_{\{E,e\}} + \sum_{R\in\mathcal{R}}\sum_{r\in\mathcal{L}_{\mathcal{R}}} c_R(r)\cdot x_{\{R,r\}} \\ & + \sum_{\substack{E_i,E_j\in\mathcal{E}\\ E_i\neq E_j}} \left[\sum_{r\in\mathcal{L}_{\mathcal{R}}}\sum_{e_1\in\mathcal{L}_{\mathcal{E}}} d^1(r,e_1)\cdot x_{\{R_{ij},r,E_i,e_1\}} + \sum_{r\in\mathcal{L}_{\mathcal{R}}}\sum_{e_2\in\mathcal{L}_{\mathcal{E}}} d^2(r,e_2)\cdot x_{\{R_{ij},r,E_j,e_2\}}\right]\end{aligned}$$

subject to:

$$\sum_{e\in\mathcal{L}_{\mathcal{E}}} x_{\{E,e\}} = 1 \qquad \forall E\in\mathcal{E} \tag{2}$$

$$\sum_{r\in\mathcal{L}_{\mathcal{R}}} x_{\{R,r\}} = 1 \qquad \forall R\in\mathcal{R} \tag{3}$$

$$x_{\{E,e\}} = \sum_{r\in\mathcal{L}_{\mathcal{R}}} x_{\{R,r,E,e\}} \qquad \forall E\in\mathcal{E} \text{ and } \forall R\in\{R : E=\mathcal{N}^1(R) \text{ or } R : E=\mathcal{N}^2(R)\} \tag{4}$$

$$x_{\{R,r\}} = \sum_{e\in\mathcal{L}_{\mathcal{E}}} x_{\{R,r,E,e\}} \qquad \forall R\in\mathcal{R} \text{ and } \forall E=\mathcal{N}^1(R) \text{ or } E=\mathcal{N}^2(R) \tag{5}$$

$$x_{\{E,e\}} \in \{0,1\} \qquad \forall E\in\mathcal{E}, e\in\mathcal{L}_{\mathcal{E}} \tag{6}$$

$$x_{\{R,r\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_{\mathcal{R}} \tag{7}$$

$$x_{\{R,r,E,e\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_{\mathcal{R}},\ E\in\mathcal{E}, e\in\mathcal{L}_{\mathcal{E}} \tag{8}$$

Figure 1: Integer Linear Programming Formulation

3.1 Linear Programming Relaxation (LPR)

To solve an ILP problem, a natural idea is to *relax* the integral constraints. That is, replacing (6), (7), and (8) with:

$$x_{\{E,e\}} \geq 0 \qquad \forall E\in\mathcal{E}, e\in\mathcal{L}_{\mathcal{E}} \tag{9}$$

$$x_{\{R,r\}} \geq 0 \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_{\mathcal{R}} \tag{10}$$

$$x_{\{R,r,E,e\}} \geq 0 \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_{\mathcal{R}}, E\in\mathcal{E}, e\in\mathcal{L}_{\mathcal{E}} \tag{11}$$

If LPR returns an integer solution, then it is also the optimal solution to the ILP problem. If the solution is non integer, then at least it gives a lower bound to the value of the cost function, which can be used in modifying the problem and getting closer to deriving an optimal integer solution. A direct way to handle the non integer solution is called *rounding*, which finds an integer point that is close to the non integer solution. Under some conditions of cost functions, which do not hold here, a well designed rounding algorithm can be shown that the rounded solution is a good approximation to the optimal solution (Kleinberg and Tardos, 1999; Chekuri et al., 2001). Nevertheless, in general, the outcomes of the rounding procedure may not even be a legal solution to the problem.

3.2 Branch & Bound and Cutting Plane

Branch and bound is the method that divides an ILP problem into several LP subproblems. It uses LPR as a subroutine to generate dual (upper and lower) bounds to reduce the search space, and finds the optimal solution as well. When LPR finds a non integer solution, it splits the problem on the non integer variable. For example, suppose variable x_i is fractional in an non integer solution to the ILP problem $\min\{cx : x\in S, x\in\{0,1\}^n\}$, where S is the linear constraints. The ILP problem can be split into two sub LPR problems, $\min\{cx : x\in S\cap\{x_i=0\}\}$ and $\min\{cx : x\in S\cap\{x_i=1\}\}$. Since any feasible solution provides an upper bound and any LPR solution generates a lower bound, the search tree can be effectively cut.

Another strategy of dealing with non integer points, which is often combined with *branch & bound*, is called *cutting plane*. When a non integer solution is given by LPR, it adds a new linear constraint that makes the non integer point infeasible, while still keeps the optimal integer solution in the feasible region. As a result, the feasible region is closer to the ideal polyhedron, which is the convex hull of feasible integer solutions. The most famous cutting plane algorithm is Gomory's fractional cutting plane method (Wolsey, 1998), which can be shown that only finite number of additional constraints are needed. Moreover, researchers develop different cutting plane algorithms for different types of ILP problems. One example is (Wang and Regan, 2000), which only focuses on binary ILP problems.

Although in theory, a search based strategy may need several steps to find the optimal solution, LPR always generates integer solutions in our experiments. This phenomenon may link to the theory of *unimodularity*.

3.3 Unimodularity

When the coefficient matrix of a given linear program in its standard form is *unimodular*, it can be shown that the optimal solution to the linear program is in fact integral (Schrijver, 1986). In other words, LPR is guaranteed to produce an integer solution.

Definition 3.1 *A matrix* $\mathbf{A}$ *of rank* m *is called* unimodular *if all the entries of* $\mathbf{A}$ *are integers, and the determinant of every square submatrix of* $\mathbf{A}$ *of order* m *is in 0,+1,-1.*

Theorem 3.1 (Veinott & Dantzig) *Let* $\mathbf{A}$ *be an* (m, n)*-integral matrix with full row rank* m*. Then the polyhedron* $\{\mathbf{x}|\mathbf{x} \geq 0; \mathbf{A}\mathbf{x} = \mathbf{b}\}$ *is integral for each integral vector* $\mathbf{b}$*, if and only if* $\mathbf{A}$ *is unimodular.*

Theorem 3.1 indicates that if a linear programming problem is in its standard form, then regardless of the cost function and the integral vector $\mathbf{b}$, the optimal solution is an integer if and only if the coefficient matrix $\mathbf{A}$ is unimodular.

Although the coefficient matrix in our problem is not unimodular, LPR still produces integer solutions for *all* the (thousands of cases) we have experimented with. This may be due to the fact that the coefficient matrix shares many properties of a unimodular matrix. As a result, most of the vertices of the polyhedron are integer points. Another possible reason is that given the cost function we have, the optimal solution is always integer. Because of the availability of very efficient LP/ILP packages, we defer the exploration of this direction for now.

4 Experiments

We describe below two experiments on the problem of simultaneously recognizing entities and relations. In the first, we view the task as a knowledge acquisition task – we let the system read sentences and identify entities and relations among them. Given that this is a difficult task which may require quite often information beyond the sentence, we consider also a "forced decision" task, in which we simulate a question answering situation – we ask the system, say, "who killed whom" and evaluate it on identifying correctly the relation and its arguments, given that it is known that somewhere in this sentence this relation is active. In addition, this evaluation exhibits the ability of our approach to incorporate task specific constraints at decision time.

Our experiments are based on the TREC data set (which consists of articles from WSJ, AP, etc.) that we annotated for named entities and relations. In order to effectively observe the interaction between relations and entities, we picked 1437 sentences that have at least one active relation. Among those sentences, there are 5336 entities, and 19048 pairs of entities (binary relations). Entity labels include 1685 *persons*, 1968 *locations*, 978 *organizations* and 705 *others*. Relation labels include 406 *located_in*, 394 *work_for*, 451 *orgBased_in*, 521 *live_in*, 268 *kill*, and 17007 *none*. Note that most pairs of entities have no active relations at all. Therefore, relation *none* significantly outnumbers others. Examples of each relation label and the constraints between a relation variable and its two entity arguments are shown as follows.

Relation	Entity1	Entity2	Example
located_in	loc	loc	(New York, US)
work_for	per	org	(Bill Gates, Microsoft)
orgBased_in	org	loc	(HP, Palo Alto)
live_in	per	loc	(Bush, US)
kill	per	per	(Oswald, JFK)

In order to focus on the evaluation of our inference procedure, we assume the problem of *segmentation* (or *phrase detection*) (Abney, 1991; Punyakanok and Roth, 2001) is solved, and the entity boundaries are given to us as input; thus we only concentrate on their classifications.

We evaluate our LP based global inference procedure against two simpler approaches and a third that is given more information at learning time. **Basic**, only tests our entity and relation classifiers, which are trained independently using only local features. In particular, the relation classifier does not know the labels of its entity arguments, and the entity classifier does not know the labels of relations in the sentence either. Since basic classifiers are used in all approaches, we describe how they are trained here.

For the entity classifier, one set of features are extracted from words within a size 4 window around the target phrase. They are: (1) words, part-of-speech tags, and conjunctions of them; (2) bigrams and trigrams of the mixture of words and tags. In addition, some other features are extracted from the target phrase, including:

symbol	explanation
icap	the first character of a word is capitalized
acap	all characters of a word are capitalized
incap	some characters of a word are capitalized
suffix	the suffix of a word is "ing", "ment", etc.
bigram	bigram of words in the target phrase
len	number of words in the target phrase
place[3]	the phrase is/has a known place's name
prof[3]	the phrase is/has a professional title (e.g. Lt.)
name[3]	the phrase is/has a known person's name

For the relation classifier, there are three sets of features: (1) features similar to those used in the entity classification are extracted from the two argument entities of

[3]We collect names of famous places, people and popular titles from other data sources in advance.

Pattern	Example
arg_1 , arg_2	San Jose, CA
arg_1 , $\cdots$ a $\cdots$ arg_2 *prof*	John Smith, a Starbucks manager $\cdots$
in/at arg_1 in/at/, arg_2	Officials in Perugia in Umbria province said $\cdots$
arg_2 *prof* arg_1	CNN reporter David McKinley $\cdots$
arg_1 $\cdots$ native of $\cdots$ arg_2	Elizabeth Dole is a native of Salisbury, N.C.
arg_1 $\cdots$ based in/at arg_2	Leslie Kota, a spokeswoman for K mart based in Troy, Mich. said $\cdots$

Table 1: Some patterns used in relation classification

the relation; (2) conjunctions of the features from the two arguments; (3) some patterns extracted from the sentence or between the two arguments. Some features in category (3) are "the number of words between arg_1 and arg_2 ", "whether arg_1 and arg_2 are the same word", or "arg_1 is the beginning of the sentence and has words that consist of all capitalized characters", where *arg*$_1$ and *arg*$_2$ represent the first and second argument entities respectively. In addition, Table 1 presents some patterns we use.

The learning algorithm used is a variation of the Winnow update rule incorporated in SNoW (Roth, 1998; Roth and Yih, 2002), a multi-class classifier that is specifically tailored for large scale learning tasks. SNoW learns a sparse network of linear functions, in which the targets (entity classes or relation classes, in this case) are represented as linear functions over a common feature space. While SNoW can be used as a classifier and predicts using a winner-take-all mechanism over the activation value of the target classes, we can also rely directly on the raw activation value it outputs, which is the weighted linear sum of the active features, to estimate the posteriors. It can be verified that the resulting values are monotonic with the confidence in the prediction, therefore provide a good source of probability estimation. We use softmax (Bishop, 1995) over the raw activation values as conditional probabilities. Specifically, suppose the number of classes is n, and the raw activation values of class i is act_i. The posterior estimation for class i is derived by the following equation.

$$p_i = \frac{e^{act_i}}{\sum_{1 \leq j \leq n} e^{act_j}}$$

Pipeline, mimics the typical strategy in solving complex natural language problems – separating a task into several stages and solving them sequentially. For example, a named entity recognizer may be trained using a different corpus in advance, and given to a relation classifier as a tool to extract features. This approach first trains an entity classifier as described in the *basic* approach, and then uses the prediction of entities in addition to other local features to learn the relation identifier. Note that although the true labels of entities are known here when training the relation identifier, this may not be the case in general NLP problems. Since only the *predicted* entity labels are available in testing, learning on the predictions of the entity classifier presumably makes the relation classifier more tolerant to the mistakes of the entity classifier. In fact, we also observe this phenomenon empirically. When the relation classifier is trained using the true entity labels, the performance is much worse than using the predicted entity labels.

LP, is our global inference procedure. It takes as input the constraints between a relation and its entity arguments, and the output (the estimated probability distribution of labels) of the basic classifiers. Note that *LP* may change the predictions for either entity labels or relation labels, while *pipeline* fully trusts the labels of entity classifier, and only the relation predictions may be different from the basic relation classifier. In other words, *LP* is able to enhance the performance of entity classification, which is impossible for *pipeline*.

The final approach, **Omniscience**, tests the conceptual upper bound of this entity/relation classification problem. It also trains the two classifiers separately as the *basic* approach. However, it assumes that the entity classifier knows the correct relation labels, and similarly the relation classifier knows the right entity labels as well. This additional information is then used as features in training and testing. Note that this assumption is totally unrealistic. Nevertheless, it may give us a hint that how much a global inference can achieve.

4.1 Results

Tables 2 & 3 show the performance of each approach in $F_{\beta=1}$ using 5-fold cross-validation. The results show that *LP* performs consistently better than *basic* and *pipeline*, both in entities and relations. Note that *LP* does not apply learning at all, but still outperforms *pipeline*, which uses entity predictions as new features in learning. The results of the *omniscient* classifiers reveal that there is still room for improvement. One option is to apply learning to tune a better cost function in the *LP* approach.

One of the more significant results in our experiments, we believe, is the improvement in the *quality* of the decisions. As mentioned in Sec. 1, incorporating constraints helps to avoid inconsistency in classification. It is in-

Approach	person			organization			location		
	Rec.	Prec.	F_1	Rec.	Prec.	F_1	Rec.	Prec.	F_1
Basic	89.4	89.2	89.3	86.9	91.4	89.1	68.2	90.9	77.9
Pipeline	89.4	89.2	89.3	86.9	91.4	89.1	68.2	90.9	77.9
LP	90.4	90.0	90.2	88.5	91.7	90.1	71.5	91.0	80.1
Omniscient	94.9	93.5	94.2	92.3	96.5	94.4	88.3	93.4	90.8

Table 2: Results of Entity Classification

Approach	located_in			work_for			orgBased_in		
	Rec.	Prec.	F_1	Rec.	Prec.	F_1	Rec.	Prec.	F_1
Basic	54.7	43.0	48.2	42.1	51.6	46.4	36.1	84.9	50.6
Pipeline	51.2	51.6	51.4	41.4	55.6	47.5	36.9	76.6	49.9
LP	53.2	59.5	56.2	40.4	72.9	52.0	36.3	90.1	51.7
Omniscient	64.0	54.5	58.9	50.5	69.1	58.4	50.2	76.7	60.7

Approach	live_in			kill		
	Rec.	Prec.	F_1	Rec.	Prec.	F_1
Basic	39.7	61.6	48.3	82.1	73.6	77.6
Pipeline	42.6	62.2	50.6	83.2	76.4	79.6
LP	41.5	68.1	51.6	81.3	82.2	81.7
Omniscient	57.0	60.7	58.8	82.1	74.6	78.2

Table 3: Results of Relation Classification

teresting to investigate how often such mistakes happen without global inference, and see how effectively the global inference enhances this.

For this purpose, we define the *quality* of the decision as follows. For an active relation of which the label is classified correctly, if both its argument entities are also predicted correctly, we count it as a *coherent* prediction. *Quality* is then the number of *coherent* predictions divided by the sum of *coherent* and *incoherent* predictions. Since the *basic* and *pipeline* approaches do not have a global view of the labels of entities and relations, 5% to 25% of the predictions are incoherent. Therefore, the quality is not always good. On the other hand, our global inference procedure, LP, takes the natural constraints into account, so it never generates incoherent predictions. If the relation classifier has the correct entity labels as features, a good learner should learn the constraints as well. As a result, the quality of *omniscient* is almost as good as *LP*.

Another experiment we did is the *forced decision* test, which boosts the F_1 of "kill" relation to 86.2%. Here we consider only sentences in which the "kill" relation is active. We force the system to determine which of the possible relations in a sentence (i.e., which pair of entities) has this relation by adding a new linear equality. This is a realistic situation (e.g., in the context of question answering) in that it adds an external constraint, not present at the time of learning the classifiers and it evaluates the ability of our inference algorithm to cope with it. The results exhibit that our expectations are correct. In fact, we believe that in natural situations the number of constraints that can apply is even larger. Observing the algorithm performs on other, specific, forced decision tasks verifies that LP is reliable in these situations. As shown in the experiment, it even performs better than *omniscience*, which is given more information at learning time, but cannot adapt to the situation at decision time.

5 Discussion

We presented an linear programming based approach for global inference where decisions depend on the outcomes of several different but mutually dependent classifiers. Even in the presence of a fairly general constraint structure, deviating from the sequential nature typically studied, this approach can find the optimal solution efficiently.

Contrary to general search schemes (e.g., beam search), which do not guarantee optimality, the linear programming approach provides an efficient way to finding the optimal solution. The key advantage of the linear programming formulation is its generality and flexibility; in particular, it supports the ability to incorporate classifiers learned in other contexts, "hints" supplied and decision time constraints, and reason with all these for the best global prediction. In sharp contrast with the typically used pipeline framework, our formulation does not blindly trust the results of some classifiers, and therefore is able to overcome mistakes made by classifiers with the help of constraints.

Our experiments have demonstrated these advantages by considering the interaction between entity and relation classifiers. In fact, more classifiers can be added and used within the same framework. For example, if coreference resolution is available, it is possible to incorporate it in the form of constraints that force the labels of the co-referred entities to be the same (but, of course, allowing the global solution to reject the suggestion of these classifiers). Consequently, this may enhance the performance of entity/relation recognition and, at the same time, correct possible coreference resolution errors. Another example is to use chunking information for better relation identification; suppose, for example, that we have available chunking information that identifies Subj+Verb and Verb+Object phrases. Given a sentence that has the verb “murder”, we may conclude that the subject and object of this verb are in a “kill” relation. Since the chunking information is used in the global inference procedure, this information will contribute to enhancing its performance and robustness, relying on having more constraints and overcoming possible mistakes by some of the classifiers. Moreover, in an interactive environment where a user can supply new constraints (e.g., a question answering situation) this framework is able to make use of the new information and enhance the performance at decision time, without retraining the classifiers.

As we show, our formulation supports not only improved accuracy, but also improves the ‘human-like” quality of the decisions. We believe that it has the potential to be a powerful way for supporting natural language inferences.

Acknowledgements This research has been supported by NFS grants CAREER IIS-9984168, ITR IIS-0085836, EIA-0224453, an ONR MURI Award, and an equipment donation from AMD. We also thank the anonymous referees for their useful comments.

References

S. Abney. 1991. Parsing by chunks. In S. Abney R. Berwick and C. Tenny, editors, *Principle-based parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer, Dordrecht.

C. Bishop, 1995. *Neural Networks for Pattern Recognition*, chapter 6.4: Modelling conditional distributions, page 215. Oxford University Press.

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November.

C. Chekuri, S. Khanna, J. Naor, and L. Zosin. 2001. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118.

CPLEX. 2003. ILOG, Inc. CPLEX. http://www.ilog.com/products/cplex/.

T. Dietterich. 2002. Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Springer-Verlag.

J. Kleinberg and E. Tardos. 1999. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *IEEE Symposium on Foundations of Computer Science*, pages 14–23.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

A. Levin, A. Zomet, and Yair Weiss. 2002. Learning to perceive transparency from the statistics of natural scenes. In *NIPS-15; The 2002 Conference on Advances in Neural Information Processing Systems*.

S. Li. 2001. *Markov Random Field Modeling in Image Analisys*. Springer-Verlag.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. MIT Press.

D. Roth and W. Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *COLING 2002, The 19th International Conference on Computational Linguistics*, pages 835–841.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*, pages 806–813.

D. Roth. 2002. Reasoning with classifiers. In *Proc. of the European Conference on Machine Learning*, pages 506–510.

A. Schrijver. 1986. *Theory of Linear and Integer Programming*. Wiley Interscience series in discrete mathmatics. John Wiley & Sons, December.

B. Taskar, A. Pieter, and D. Koller. 2002. Discriminative probabilistic models for relational data. In *Proc. of Uncertainty in Artificial Intelligence*, pages 485–492.

E. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL-2003*, pages 142–147. Edmonton, Canada.

X. Wang and A. Regan. 2000. A cutting plane method for integer programming problems with binary variables. Technical Report UCI-ITS-WP-00-12, University of California, Irvine.

L. Wolsey. 1998. *Integer Programming*. John Wiley & Sons, Inc.

Xpress-MP. 2003. Dash Optimization. Xpress-MP. http://www.dashoptimization.com/products.html.