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

The typical processing paradigm in natural language processing is the "pipeline" approach, where learners are being used at one level, their outcomes are being used as features for a second level of predictions and so one. In addition to accumulating errors, it is clear that the sequential processing is a crude approximation to a process in which interactions occur across levels and down stream decisions often interact with previous decisions.

This work develops a general approach to inference over the outcomes of predictors in the presence of general constraints. It allows breaking away from the pipeline paradigm by performing global inference over the outcome of different predictors — potentially learned and evaluated given only partial information — along with domain and task specific constraints on the outcomes of the predictors. At the inference level, the existence of mutual constraints on simultaneous outcomes of predictors results in modifying these predictions to optimize global and task specific constraints.

We develop a linear programming formulation for this problem and evaluate it in the context of simultaneously learning named entities and relations between. 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 with respect to the input. These predictions need to respect some constraints that could arise from the nature of the data or from domain or task specific conditions, hence require a level of inference on top the predictions. As an example from the visual processing domain that exemplifies this point, consider the problem of counting the number of people in an image, where people can be partly occluded; a large number of "body part detectors" and scene interpretations predictors come into play, and some inference procedures takes these into account in making the final decision. Similarly, interpreting natural language sentences requires a multitude of abstractions and context dependent disambiguations that depend on each other in intricate ways. Efficient solutions to problems of these sort have been given when the constraints on the predictors are *sequential*. Variations of HMMs, conditional models and sequential variations of random Markov fields all provide efficient solutions [1, 2].

However, in many important situations, the structure of the problem is more general, resulting in a computationally intractable problem. Problems of these sorts have been studied in computer vision, where inference is typically done over low level measurements rather than over higher level predictors [3, 4]. In the context of natural language, the typical processing paradigm is the "pipeline" approach, where learners are being used at one level, and their outcomes are being used as features for a second level of predictions and so one. For example, it is typical to learn a *part-of-speech* tagger, evaluate it and use the outcome as features in the

*This research is supported by NSF grants ITR-IIS-0085836, ITR-IIS-0085980 and IIS-9984168 and an ONR MURI Award.

task of learning to identify *phrases*, say, then use all previous outcomes as features when learning, semantic predicates such as *name entities* and finally use all these when learning to identify *relations* between entities. Sometimes a sequential type approach is being used within the pipeline paradigm, when a Viterbi-like algorithm can be used [5]. In addition to accumulating errors, it is clear that the sequential processing is a crude approximation to a process in which interactions occur across levels and down stream decisions often interact with previous decisions.

This work develops a general approach to inference over the outcomes of predictors in the presence of general constraints. It allows breaking away from the pipeline paradigm by performing global inference over the outcome of different predictors — potentially learned and evaluated given only partial information — along with domain and task specific constraints on the outcomes of the predictors. At the inference level, the existence of mutual constraints on simultaneous outcomes of predictors results in modifying these predictions to optimize global and task specific constraints.

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

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. 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 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. The key insight to the technical solution we suggest comes from recent techniques developed in the context of approximation algorithms [6]. Following this work, we develop a linear programming formulation and show how to cast our problem in it. However, we still need an integral solution, and this formalisms does not guarantee it. In general, there is a need to resort to rounding methods that do not necessarily satisfy the constraints. It turns out, however, that despite the general nature of the our problem – its graph structure represents a collection of binary relations and their arguments, along with constraints on the binary relations between arguments and the relation types – the optimal solution is always integer. While we are not able to prove that this is always the case, we have developed some theoretical understanding of it; and, our experimental results are very decisive – we *always* get an optimal solution that has integral values.

When evaluated on simultaneous learning of name 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, that no human ever makes, as we show, our approach improves also the *quality* of the inference significantly.

Our approach is similar in nature, although different in its technical approach and generality to other approaches that attempt to learn several different classifiers and derive global decisions by inference over their outcomes [1, 7]. It could be contrasted with other approaches to sequential inference or to general Markov random field approaches [2, 8]. 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. We believe this to be the right conceptual framework, given the motivating applications

in NLP and computer vision described above. This way, our technical 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.

2 The Relational Inference Problem

We consider the relational inference problem within the *reasoning with classifiers* paradigm [9]. This paradigm investigates decisions that depend on the outcomes of several different but mutually dependent classifiers. The classifiers' outcomes need to respect some constraints that could arise from the sequential nature of the data or other domain specific conditions, thus requiring a level of inference on top the predictions. In this way, variables considered here are in fact outcomes of learned classifiers, learned over a large number of variables (features) that are abstracted away here, since they are not of interest for the inference process. All the information in these is contained in the classifiers' outcome.

We 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 sets $\mathcal{V}$ 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 $\mathcal{R} = \{R_{ij}\}_{\{1 \le i,j \le n; i \ne j\}}$ is viewed as binary relations over $\mathcal{E}$. Specifically, for each pair of entities E_i and E_j, $i \ne j$, 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 variables, 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 $\mathcal{V}$. $\mathcal{C}^1 : \mathcal{L}_\mathcal{E} \times \mathcal{L}_\mathcal{R} \to \{0, 1\}$ constraints values by the first argument of relations. $\mathcal{C}^2$ 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 $\mathcal{C}^R : \mathcal{L}_\mathcal{R} \times \mathcal{L}_\mathcal{R} \to \{0, 1\}$ which constrain types of relations, etc. In fact, as will be clear in Sec. 3 the language for defining constraints is very rich – linear equations over $\mathcal{V}$.

We exemplify the framework using the problem of simultaneous recognition or named entities and relations in sentences. Briefly, 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, satisfying, as best as possible the recommendation of the entity and relation classifiers, but also satisfying 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 entity 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 [1], 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 here is specifically to consider interacting problems at different levels, resulting in a 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 [10]. Rather than doing that, for computational reasons, we first use a fairly standard transformation of MRF to a discrete optimization problem (see [11] for details). Specifically, under weak assumptions we can view the inference problem as the following optimization problem, which aims to to minimize the objective function that is the sum of the following two cost functions.

Assignment cost: The costs 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 $-\log p$.

Constraints 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 ∞ [1]. Similarly, we use d^2 to force the consistency of the second argument of a relation.

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

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

3 A Computational Approach to Relational Inference

Unfortunately, it is not hard to see that the optimization problem 1 is computationally intractable even when placing assumptions on the cost function [11].

The computational approach we adopt is based on a *linear programming* formulation of the problem. We first provide an integer linear programming formulation to Eq. 1, and then *relax* it to a linear programming problem. This Linear Programming Relaxation (LPR) [12, 13] technique, in general, might find non-integer solutions. Therefore, to "round" the solutions to integer solutions is needed. Under some assumptions on the cost function, which do not hold in our case, there exist rounding procedures that guarantee some optimality [11, 6]. However, such rounding procedures do not always exist. We discuss the issue of integer solutions to linear programs and provide evidence that for our target problems, rounding is not required – the linear program always has an optimal integer solution.

Our linear programming formulation is based on the methods proposed by [6]. Since our objective function (Eq.1) is not a linear function in terms of the labels, we introduce new binary decision 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 integer programming problem.

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

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

subject to:

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

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

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

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

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

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

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

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.

To apply linear programming, we relax the *integral* constraints. That is, replacing (6), (7), and (8) with:

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

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

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

Now, we can solve our optimization problem efficiently. Obviously, in general, the solution found by the linear programming solver may not be integral. Therefore, a rounding procedure may be needed in order to generate an integer solution. The problem is that, in general, the outcomes of the rounding procedure may not be a legal solution to the problem, although under some conditions that do not hold here, it can be shown that the rounded solution is a good approximation to the optimal solution [6].

Instead of studying rounding, we take a different route and study the theory of integer solutions to linear programs. It turns out, that under some conditions on the coefficient matrix of the linear program in it canonical form, it can be shown that the optimal solution to the linear program is in fact integral. [13] Motivated by the decisive results of our experimental study (Sec. 4), we have investigated this direction.

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* $\{\mathbf{x}|\mathbf{x} \geq 0; \mathbf{Ax} = \mathbf{b}\}$ *is integral for each integral vector* $\mathbf{b}$*, if and only if* $\mathbf{A}$ *if 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 iff the coefficient matrix $\mathbf{A}$ is unimodular. The reason is that every basic feasible solution of the linear program is in the following form:

$$\frac{adj(\mathbf{B})}{||\mathbf{B}||} \cdot \mathbf{b}, \tag{12}$$

where $\mathbf{B}$ is a non-singular square submatrix of $\mathbf{A}$ of order m. Since $||\mathbf{B}||$ is either +1 or -1, and the optimal solution only appears on basic feasible solutions, this linear programming problem can only have integer solutions.

Our linear programming formulation is in the standard form, but the coefficient matrix does not have full row rank. Let n be the number of entities, l_e be the number of entity labels, and l_r be the number of relation labels. We can eliminate some rows by the following theorem.

Theorem 3.2 *The coefficient matrix* $\mathbf{A}$ *of our linear programming formulation is an* $(n^2 + 2n(n-1)(l_e + l_r), nl_e + n(n-1)l_r + 2n(n-1)l_r l_e)$ *matrix, and all the entries are in (0,+1,-1). In addition, rank(*$\mathbf{A}$*) is* $n^2 + 2n(n-1)(l_e + l_r - 1)$ *and the full rank matrix* $\mathbf{A}'$ *can be derived by eliminating the first* $n^2 - 1$ *rows and the last* $(n-1)^2$ *rows of* $\mathbf{A}$.

After this elimination, the corresponding vector $\mathbf{b}'$ in our linear programming formulation has a nice property – the first entry is 1 and all other entries are 0.

Combining the above results, we know that if the linear programming formulation has any of the following properties, we will always get an optimal integer solution. (1) $\mathbf{A}'$ is unimodular. (2) The first column of $adj(\mathbf{B}')$ is $(a_1, a_2, \ldots, a_r)^T ||\mathbf{B}'||$, where $\mathbf{B}'$ is a non singular submatrix of $\mathbf{A}'$ of order = rank($\mathbf{A}'$), r = rank($\mathbf{A}$) and all $a_i, 1 \leq i \leq r$, are integers. (3) The constraints in our problem forbid non integer solutions.

We find many of our problems have at least one of the above properties. For those that don't have the properties, by examining the coefficient matrix, we observe that the non integer solution can only happen when there are at least two integer solutions that also have the optimal cost. Although at this point we are still looking for the proof, we have the following conjecture.

Conjecture 3.1 *Any non integer basic feasible solution* $\mathbf{x}$ *in our problem is a convex combination of several integer basic feasible solutions* $\mathbf{y_1}, \ldots, \mathbf{y_k}$.

Since the cost function c, is a convex function (linear), there exists a $y_i (1 \leq i \leq k)$ such that the $c(\mathbf{x}) \geq c(\mathbf{y_i})$. Because $\mathbf{x}$ is optimal, therefore $c(\mathbf{x}) = c(\mathbf{y_i})$. Note that in practice, it is very unlikely to have two optimal solutions. As mentioned above, none of the thousand cases we experimented requires rounding.

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 a 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 pair of entities have no active relation 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*) [14, 1] is solved, and the entity boundaries are given to us as input; thus we only concentrate on their classification.

We evaluate our linear programming based global inference procedure, **LPR** 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. Before moving on to describe the other approaches, we describe the basic classifiers, used in all the approaches.

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[2]	the phrase is/has a known place's name
prof[2]	the phrase is/has a professional title (e.g. Lt.)
name[2]	the phrase is/has a known person's name

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

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 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 [15, 16], 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 [17] 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.

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

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

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 when training the relation identifier, this may not be the case in general NLP problems. Therefore, the predictions of the entity classifier are used instead.

LPR, 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 *LPR* 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, *LPR* 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

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
LPR	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	Q.	Rec.	Prec.	F_1	Q.	Rec.	Prec.	F_1	Q.
Basic	54.7	43.0	48.2	89.2	42.1	51.6	46.4	74.1	36.1	84.9	50.6	93.6
Pipeline	51.2	51.6	51.4	88.9	41.4	55.6	47.5	76.7	36.9	76.6	49.9	94.0
LPR	53.2	59.5	56.2	100	40.4	72.9	52.0	100	36.3	90.1	51.7	100
Omniscient	64.0	54.5	58.9	100	50.5	69.1	58.4	100	50.2	76.7	60.7	100

Approach	live_in				kill			
	Rec.	Prec.	F_1	Q.	Rec.	Prec.	F_1	Q.
Basic	39.7	61.6	48.3	92.3	82.1	73.6	77.6	96.8
Pipeline	42.6	62.2	50.6	90.5	83.2	76.4	79.6	96.9
LPR	41.5	68.1	51.6	100	81.3	82.2	81.7	100
Omniscient	57.0	60.7	58.8	100	82.1	74.6	78.2	99.5

Table 3: Results of Relation Classification

Tables 2 & 3 show the performance of each approach in $F_{\beta=1}$. The results show that *LPR* performs consistently better than *basic* and *pipeline*, both in entities and relations. While we do not show it for lack of space, it enhances both recall and precision, reflected in a F_1. Note that *LPR* 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 *LPR* 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 *stupid* (from a human perspective) mistakes in classification. It is interesting to examine 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*(Q.) 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, a certain amount of the predictions are incoherent. Therefore, the quality is not always good. On the other hand, our global inference procedure, LPR, 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 *LPR*.

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, and we force the system to determine which of the possible relations in a sentence (i.e., which pair of entities) has this relation. 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 constraint that can apply is even larger, and observing the algorithm performs on other, specific, forced decision tasks verifies that LPR is reliable in these situation, and as shown, performs better than *omniscience* who is given more information at learning time, but cannot adapt to the situation at decision time.

5 Discussion

We presented an approach for global inference in situations where decisions depend on the outcomes of several different but mutually dependent classifiers whose outcomes need to respect domain or tasks specific constraint.

We have shown that even in the presence of a fairly general constraint structure, deviating from the sequential nature typically studied, we can develop an optimal and efficient inference procedure. Our technical approach is based on formulating the discrete inference problem as a linear program, and arguing that it admits an integer solution. Although we have developed some theoretical understanding, our claim is based so far only on overwhelming experimental evidence, and an attempt to solidify it is on top of our agenda.

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.

References

[1] V. Punyakanok and D. Roth. 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, 2001.

[2] J. Lafferty, A. McCallum, and F. Pereira. 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, 2001.

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

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

[5] A. Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

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

[7] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. Named entity recognition with character-level models. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 180–183. Edmonton, Canada, 2003.

[8] B. Tasker, A. Pieter, and D. Koller. Discriminative probabilistic models for relational data. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference (UAI-2002)*, pages 485–492, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

[9] D. Roth. Reasoning with classifiers. In *Proc. of the European Conference on Machine Learning*, pages 506–510, 2001.

[10] S. Li. *Markov Random Field Modeling in Image Analisys*. Springer-Verlag, 2001.

[11] J. Kleinberg and E. Tardos. 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, 1999.

[12] G. Sierksma. *Linear and integer programming : theory and practice*. Marcel Dekker, January 2001.

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

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

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

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

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