Markov Logic Networks for Text Mining: A Qualitative and Empirical Comparison with Integer Linear Programming

Luis Gerardo Mojica and Vincent Ng
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, TX 75083-0688, USA
{mojica,vince}@hlt.utdallas.edu

Abstract

Joint inference approaches such as Integer Linear Programming (ILP) and Markov Logic Networks (MLNs) have recently been successfully applied to many natural language processing (NLP) tasks, often outperforming their pipeline counterparts. However, MLNs are arguably much less popular among NLP researchers than ILP. While NLP researchers who desire to employ these joint inference frameworks do not necessarily have to understand their theoretical underpinnings, it is imperative that they understand which of them should be applied under what circumstances. With the goal of helping NLP researchers better understand the relative strengths and weaknesses of MLNs and ILP; we will compare them along different dimensions of interest, such as expressiveness, ease of use, scalability, and performance. To our knowledge, this is the first systematic comparison of ILP and MLNs on an NLP task.

Keywords: Markov Logic, Integer Linear Programming, Joint Inference

1. Introduction

In the early days of the statistical natural language processing (NLP) era, many language processing tasks were tackled using the so-called *pipeline* architecture: the given task is broken into a series of sub-tasks such that the output of one sub-task is an input to the next sub-task in the sequence. This pipeline architecture is appealing for various reasons, including modularity, modeling convenience, and manageable computational complexity. However, it suffers from the *error propagation* problem: errors made in one sub-task are propagated to the next sub-task in the sequence.

Realizing this weakness, researchers have turned to *joint inference* approaches such as Integer Linear Programming (ILP) and Markov Logic Networks (MLNs), which often outperform their pipeline counterparts. These joint inference approaches enable manual specification of constraints. These constraints effectively allow incorporation of background knowledge into NLP systems, addressing the aforementioned error propagation problem by allowing the downstream components to influence the upstream components in a pipelined system architecture. To date, MLNs have been underused in NLP applications, and are arguably much less popular among NLP researchers than ILP. While NLP researchers who desire to employ these joint inference frameworks do not necessarily have to understand their theoretical underpinnings, it is imperative that they understand which of them should be applied under what circumstances.

In light of this discussion, our goal in this paper is to help NLP researchers better understand the relative strengths and weaknesses of MLNs and ILP by comparing them along different dimensions of interest, such as *expressiveness* (what can and cannot be expressed using a specification language?), *ease of use* (how easy is it to encode constraints?), *scalability* (how efficient is the inference procedure, especially when applied to large problems?), and *performance* (how well do these inference algorithms perform?).

To facilitate the comparison, we will discuss how ILP and MLNs can be applied to the task of *fine-grained opinion extraction*. While many tasks could have been chosen, we chose this task not only for its importance but also for the fact that its sophistication allows us to demonstrate the differences between the two inference frameworks. In fact, while ILP has been applied to this task (Yang and Cardie, 2013) (henceforth Y&C), MLNs have not, so it would be informative for us to consider how MLNs can be applied to it. To our knowledge, this is the first systematic comparison of ILP and MLNs on an NLP task. It is worth mentioning that while our discussion is centered on this task, many of our conclusions and recommendations are generally applicable to other NLP tasks.

The rest of this paper is organized as follows. Section 2 provides some background information, including the task of fine-grained opinion extraction, the corpus, and the two inference algorithms. Section 3 describes the constraints that we enforce on the outputs of a fine-grained opinion extraction system and how ILP and MLNs encode such constraints. In Sections 4 and 5, we discuss the relative strengths and weaknesses of MLNs and ILP. Finally, we report our empirical results in Section 6, and discuss our conclusions in Section 7.

2. Preliminaries

2.1. Fine-Grained Opinion Extraction

2.1.1. Task Definition

Fine-grained opinion extraction is an opinion mining task that involves (1) identifying text spans corresponding to *opinions* and their *arguments* and (2) the relations between them. Compared to document-level opinion mining (e.g., determining whether a customer review is positive, negative, or neutral), fine-grained opinion extraction occurs at the sentence and phrase levels and is comparatively less investigated.

This fine-grained opinion extraction task is typically decomposed into two subtasks. The first subtask, *entity extraction*, involves identifying three types of opinionated entities, including opinions ("O") and two types of arguments, Sources ("S", entities generating opinions) and Targets ("T", entities of which a opinion is about). The second subtask, *relation extraction*, involves extracting *Is from* relations (i.e., linking a source to its opinion) and *Is about* relations (i.e., linking a target to its opinion). To better understand the task, consider the following example taken from the MPQA[1] corpus.

1. *[Ashcroft]$_{S_0}$* ***told***$_{O_0}$ *us [he]$_{S_0}$* ***was determined***$_{O_1}$ *[to take every conceivable action]$_{T_1}$*

Ashcroft is the source of the two opinions, **told** and **was determined**, and *to take every conceivable action* is the target of the second opinion. In other words, there are two *Is from* relations: one between *Ashcroft* and **told** and the other between *Ashcroft* and **was determined**, and there is one *Is about* relation between **was determined** and *to take every conceivable action*. Note that the opinion O_0 has no target associated with it. When an opinion has no target arguments or source arguments, we refer to it as target-implicit or source-implicit, respectively. In this case, O_0 is target-implicit.

2.1.2. Corpus

For training and evaluation, we use the MPQA 2.0 corpus (Wiebe et al., 2005; Wilson, 2008). After discarding those ill-formatted documents (lack of punctuation, paragraphs, etc.), we obtain 433 documents with $8,377$ sentences. These documents contains $4,717$ opinions, $4,680$ targets and $5,505$ sources. The number of *Is from* relations is $13,046$, and $9,763$ of the *Is about* type. Unlike Y&C, we do *not* remove sentences containing no opinionated entities.

2.1.3. Why Joint Inference?

A straightforward method to address the fine-grained opinion extraction task is to adopt a *pipeline* approach, where we (1) use an entity extraction model to extract the opinions, sources, and targets; (2) use *argument-implicit* classifiers to identify those extracted opinions that are source-implicit as well as those that are target-implicit; and (3) use *opinion-argument* classifiers to extract *Is from* relations between sources and opinions that are not source-implicit, as well as *Is about* relations between targets and opinions that are not target-implicit.[2] However, this so-called *pipeline* approach suffers from *error propagation*, where errors made in the entity extraction model will be propagated to the implicit opinion identification component, which in turn will be propagated to the relation extraction component. For example, in the example sentence above, if the entity extraction model failed to retrieve the span [Aschroft], or if the source-implicit classifier misclassifies opinions O_0 and O_1 (as source-implicit), it would not be possible for the opinion-argument classifier to extract the *Is from* relations between this span and O_0 and O_1.

One way to address the aforementioned error propagation problem is to perform *joint inference* over the outputs of the entity extraction model, the argument-implicit classifiers, and the opinion-argument classifiers. Unlike in the pipeline approach, where entity extraction influences implicit opinion identification, which in turn influences relation extraction (but not vice versa), in a joint inference approach, all three tasks can influence each other.

Specifically, recall that one problem with the pipeline approach is that if the entity extraction model fails to extract an entity, the opinion-argument classifiers cannot extract the relations for the opinion in consideration. To be robust to the errors made by the entity extraction model, instead of making use of its *1-best* output, we make use of its *n-best* output and the confidence assigned by the model to each *candidate* entity it extracted. Similarity, to be robust to the errors made by the argument-implicit classifiers, instead of making use of its binary decisions, we make use of the confidences associated with its decisions. Given this setup, if the opinion-argument classifier is highly confident that an *Is about* relation exists between two *candidate* entities, then these two entities will likely be extracted as an opinion and a target even if the entity extraction component fails to extract them or the target-implicit classifier erroneously determines that the opinion candidate is target-implicit. In other words, the final entity extraction decisions and relation extraction decisions will be made *jointly* by the entity extraction model, the argument-implicit classifiers, and the opinion-argument classifiers by considering the confidence values they individually assign to the extraction decisions.

2.2. Joint Inference Frameworks

In this subsection, we provide a brief overview of two joint inference frameworks, ILP and MLNs.

2.2.1. Integer Linear Programming

At a high level, many NLP tasks are structured prediction problems which can be naturally expressed as constrained optimization problems, where the goal is to optimize an objective function subject to a set of linear (equality and inequality) constraints. In principle, a variety of methods can be used to solve these problems. ILP methods are arguably the most popular choice among NLP researchers. Formally, an ILP problem is defined as follows:

$$\begin{aligned} &\text{Maximize:} && f(x_1, x_2, ..., x_n) \\ &\text{Subject to:} && g_j(x_1, x_2, ..., x_n) \geq b_j \quad (j = 1, 2, ..., m) \end{aligned}$$

where x_i are the variables that take finite integer values, $f(x_1, x_2, ..., x_n)$ is the objective function, and g_j $(x_1, x_2, ..., x_n)$, $1 \leq j \leq m$, are the constraints (each constraint is linear in $x_1, x_2, ..., x_n$). Several highly optimized open source and commercial software for solving ILP problems, such as *lpsolve* (Berkelaar et al., 2004) and *Gurobi* [3],

[1] http://mpqa.cs.pitt.edu/corpora/mpqa_corpus/

[2] Our implementation of the entity extraction model, the argument-implicit classifiers and the opinion-argument classifiers follow that of Yang and Cardie (2013). Owing to space limitations, we omit the details of the training procedure as well as the features used to train each model, and refer the reader to Y&C's paper for details.

[3] www.gurobi.com

are readily available, and therefore the application designer can focus on modeling issues rather than solving optimization problems

2.2.2. Markov Logic Networks

Markov logic (Richardson and Domingos, 2006; Domingos and Lowd, 2009), a popular statistical relational learning (SRL) approach (Taskar and Getoor, 2007), combines graphical models with first-order logic. At a high level, a MLN is a set of weighted first-order logic formulas (f_i, w_i), where w_i is the weight associated with formula f_i. Given a set of constants that model objects in the domain, it defines a Markov network (Koller and Friedman, 2009) in which we have (1) one node per *atom* (i.e., grounded predicate) and (2) a propositional *feature* corresponding to each grounding of each first-order formula (i.e., a clique formed from the atoms in the formula). The weight of the feature is the weight of the corresponding first-order formula.

Formally, the probability of a world ω which represents an assignment of values to all atoms in the Markov network is given by:

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N(f_i, \omega)\right)$$

where $N(f_i, \omega)$ is the number of groundings of f_i which evaluate to True in ω and Z is a normalization constant called the partition function.

The key inference tasks over MLNs are computing the partition function (Z) and the most-probable explanation given evidence (the MAP task). Most queries can be reduced to these inference tasks. Formally, the partition function and the MAP tasks are given by:

$$Z = \sum_\omega \exp\left(\sum_i w_i N(f_i, \omega)\right) \quad (1)$$

$$\arg\max_\omega P(\omega) = \arg\max_\omega \sum_i w_i N(f_i, \omega) \quad (2)$$

Unlike ILP, which optimizes an objective function provided by the user, MLNs typically optimize the conditional likelihood of the data. Software packages such as Alchemy (Kok et al., 2008), Alchemy 2.0 (Venugopal and Gogate, 2012), Markov the beast (Riedel, 2009) and Tuffy (Niu et al., 2011) for inference and learning with MLNs are widely available.

3. Joint Inference for Fine-Grained Opinion Extraction

3.1. Consistency Constraints

As mentioned before, joint inference enables the entity extraction model, the argument-implicit classifiers, and the opinion-argument classifiers to influence each other by employing constraints to enforce *global consistency* over their outputs. In this subsection, we enumerate seven constraints. Note that Constraints **(1)** and **(2)** are *intra-task* constraints, which enforce consistency over the outputs of the entity extraction model, whereas the other constraints are *inter-task* constraints, which enforce consistency over the outputs of models for different tasks.

The first five constraints were originally proposed by Y&C. Constraint **(1)** states that an opinion/source/target *candidate* (obtained from the 30-best output of the CRF-based entity extraction model) can only be assigned exactly one of four types: opinion, target, source, or none (if it does not belong to any of the other three types). Constraint **(2)** indicates that among every pair of overlapping entity candidates, at most one should be extracted as a non-none type entity. Constraint **(3)** enforces the consistency between opinion-argument classifiers and the argument-implicit classifiers. Specifically, an opinion candidate can be related to source and target arguments if it is not an argument-implicit opinion. Constraint **(4)** enforces the consistency between the opinion-argument classifiers and the entity extractor. Specifically, the relations between opinion and arguments should be consistent with their entity types (i.e., a *Is from* relation must involve an opinion and a source, and a *Is about* relation must involve an opinion and a target). In the same way, a span related with a target or source argument must be an opinion. Constraint **(5)** enforces the consistency between the opinion-implicit classifiers and the entity extractor. Specifically, an opinion candidate that is not argument-implicit must be an opinion.

Hypothesizing that some verb senses and argument roles defined in VerbNet (Schuler, 2005) are useful for identifying text spans corresponding to opinions and their arguments as well as their relationships[4], we introduce three constraints that exploit the PropBank-style semantic role labels provided by Mate Tools (Björkelund et al., 2009). Specifically, Constraint **(6)** states that (1) a span candidate associated with a particular verb sense should be assigned a non-*none* label, and (2) a span candidate associated with a verb argument should be assigned a non-*none* label. Constraint **(7)** indicates that two entities that are assigned a verb sense and an argument role respectively might be in a *Is from* or *Is about* relation. Note that while the first five constraints are *hard* constraints, Constraints **(6)** and **(7)** are *soft* constraints.

3.2. ILP Formulation

In this subsection, we show how to formulate fine-grained opinion extraction in the ILP framework. Recall that ILP requires that we define a constrained optimization problem. Below we first define the objective function and then describe how we encode the aforementioned seven constraints.

We create one ILP program for each test sentence. Specifically, for each test sentence, let O be the set of opinion candidates (provided by the 30-best CRF output), A_k be the set of argument candidates (also provided by the 30-best CRF output), where k denotes the relation type (*Is about* or *Is from*), and S be the union of O and A_k.

[4]For example, in the sentence *Williamson still hopes for an expedited review by the report*, the opinion *hopes* is assigned the verb sense `hope.01`, and its A0 and A1 arguments, *Williamson* and *for an expedited review by the report*, are its source and target, respectively.

Num	Consistency constraint	ILP	MLN
1	Uniqueness	$\sum_z x_{iz} = 1$	$\exists_c$ `Span(i,c!).`
2	Non-Overlapping	$\sum_{z \neq N} x_{iz} + \sum_{z \neq N} x_{jz} \leq 1$	`Overlap(i,j)⇒ (Span(i,N) v Span(j,N))`
3	Explicit Relation & Implicit Relation Classifiers	[1] $\sum_{j \in A_k} u_{ij} = 1 - v_{ik} + a_{ik} + b_{ik}$ [2] $a_{ik} \leq 1 - v_{ik}$; $b_{ik} \leq 1 - v_{ik}$	`[1]Implicit_src(i)⇒ !Is_from(i,j)` `[2]Implicit_trg(i)⇒ !Is_about(i,j)`

4	Relation Classifier & Entity Extractor	[1] $\sum_{i \in O} u_{ij} = x_{jz} + c_{jk} + d_{jk}$
		[2] $c_{jk} \leq x_{jz}$; $d_{jk} \leq x_{jz}$
			`[1] Is_from(i,j)⇒Span(j,S)`
			`[2]` $\exists_i$`Span(j,S)⇒Is_from(i,j)`
			`[3] Is_about(i,j)⇒Span(j,T)`
			`[4]` $\exists_i$`Span(j,T)⇒Is_about(i,j)`
			`[5] Is_from(i,j)⇒Span(i,O)`
			`[6] Is_about(i,j)⇒Span(i,O)`

5	Implicit Classifiers & Entity Extractor	$v_{ik} + x_{iO} \geq 1$	`[1]!Implicit_src(i)⇒Span(i,O)` `[2]!Implicit_trg(i)⇒Span(i,O)`
6	SRL & Entity Extractor	–	`[1] Sense(i,s+) ⇒ Span(i,c+)` `[2] Role(i,r+) ⇒ Span(i,c+)`
7	SRL & Relation Classifier	–	`[1] Sense(i,s+) ∧ Role(j,r+) ⇒Is_from(i,j)` `[2] Sense(i,s+) ∧ Role(j,r+) ⇒Is_about(i,j)`
–	MLN Consistency	–	`[1] !Is_from(i,i).` `[2] !Is_about(i,i).`

Table 1: Consistency constraints for fine-grained opinion extraction encoded as linear constraints for ILP and first-order logic formulas for MLNs.

Next, we introduce a set of binary indicator variables whose values are to be determined by ILP during the joint inference process. Specifically, x_{iz} has the value 1 if ILP believes that span i should have entity label z, where $z =\in$ {opinion, target, source, none}; u_{ij} has the value 1 if and only if ILP believes that opinion candidate i in O has a relation with argument candidate j in A_k and v_{ik} has the value 1 if ILP believes that this opinion candidate is related to a *null* argument in the relation type k.

Finally, we combine these binary variables (x_{iz}, u_{ij} , and v_{ik}) with the confidence values returned by the entity extraction model (f_{iz}), the opinion-argument classifiers (r_{ij}), and the argument-implicit classifiers ($r_{i\emptyset}$) into the following objective function:

$$\arg\max_{x,u,v} \lambda \sum_{i \in S} \sum_{z} f_{iz} x_{iz} \tag{3}$$

$$+(1-\lambda) \sum_{k} \sum_{i \in O} \left(\sum_{j \in A_k} r_{ij} u_{ij} + r_{i\emptyset} v_{ik} \right)$$

As we can see, the function is a linear combination of the confidence values from the three predictors (f_{iz}, r_{ij}, $r_{i\emptyset}$), and λ is a parameter used to balance the contribution of the entity extraction model and the relation extraction classifiers.

The objective function will be optimized subject to the first five consistency constraints defined in the previous section. The column under "ILP" in Table 1 shows how these constraints can be encoded in ILP. The encoding of the first five constraints is due to Y&C.

Two points deserve mention. First, merely enforcing the consistency between the argument-implicit classifiers and the opinion-argument classifiers in Constraint **(3)** does not require the use of the variables a_{ik} and b_{ik}. These two variables are used to enforce an additional constraint: an opinion can be related to at most three arguments. The same is true for Constraint **(4)**: the variables c_{jk} and d_{jk} are used to enforce the additional constraint that a source or target can be related to at most three opinions. Second, we do not encode Constraints **(6)** and **(7)** as ILP constraints. The reason is that these two constraints are intended as soft constraints: although it is possible to soften hard constraints in ILP, it is by no means easy to directly encode soft constraints in ILP, as we will see in the next section.

3.3. MLN Formulation

Unlike ILP, which operates at the propositional level by defining propositional variables, MLNs employ first-order logic, which defines *predicates* that operate on sets of objects. MLNs employ two types of predicates.

Query predicates are those whose assignments are not given during inference and thus need to be predicted. For the fine-grained opinion extraction task, we define five query predicates. `Span(i,l)` is true when the label assigned to text span `i` is `l`. `Is_about(i,j)` asserts that opinion `i` is related to source `j`. Similarly, `Is_from(i,j)` asserts that opinion `i` is related to target `j`. Finally, `Implicit_src(i)` and `Implicit_trg(i)` assert that opinion `i` is source-implicit and target-implicit, respectively.

Evidence predicates are those whose values are known during inference. We define three evidence predicates. `Overlap(i,j)` indicates that spans `i` and `j` overlap. `Sense(i,s)` and `Role(i,r)` indicate that span `i` has verb sense `s` and verb argument `r` assigned to it, respectively.

The column under "MLN" in Table 1 shows how the

seven consistency constraints can be encoded in an MLN. For instance, the formula $\exists_c$ `Span(i,c!)`, which implements Constraint **(1)**, encodes the hard constraint that each span i can have exactly one label. (The `!` symbol asserts that the labels assigned to a span are mutually exclusive.) The formulas `Sense(i,s+)` $\Rightarrow$ `Span(i,c+)` and `Role(i,r+)` $\Rightarrow$ `Span(i,c+)`, which implement Constraint **(6)**, encode the soft constraint that a candidate entity having sense s or role r should be labeled as a span with entity type c. The (+) operator asserts that there exists an instantiated formula for every combination of values in the variable domain. For instance, given `Sense(i,s+)` $\Rightarrow$ `Span(i,c+)`, one formula will be instantiated for each combination of sense s and entity label c of candidate span i.

In addition to the seven constraints, we need the two formulas shown in the last row of Table 1. They ensure that spans are not related to themselves. Note that this constraint needs to be explicitly specified for the MLN but not for ILP. The reason is that MLN operates at the predicate level, meaning that it will instantiate all predicates within its arguments domain. On the other hand, in ILP, we have control over which variables are created because ILP operates at the propositional level.

As mentioned before, MLN formulas encode either hard or soft constraints. Hard constraints are encoded as formulas with infinite weights (those ending with a period), whereas soft constraints are encoded as formulas with finite weights (i.e., when a soft formula logically evaluates to True, the knowledge encoded in it is proportional to the their weight). Note that while Constraints **(2)**−**(5)** are originally intended as hard constraints, they are implemented as soft constraints in our MLN. The reason is that doing so may make the MLN more robust to the noise inherent in the entity extractor and the relation classifiers' classifications.

A few other points deserve mention. First, while ILP incorporates the confidence values associated with the various classifiers' outputs into the objective function, these outputs are incorporated into an MLN as *soft evidence*, which can be thought of as our *prior* belief that a given atom (i.e., a grounded query predicate) is true. Specifically, we include as priors the atoms `Span(s,l` $\neq$ `N)` with weight $\mathtt{w}_e$ when $p(s = l|\mathbf{x}) \geq \gamma$, where $p(s = l|\mathbf{x})$ is the probability that the CRF thinks span s has entity type l given evidence $\mathbf{x}$. We include another atom `Span(s,N)` with weight $\mathtt{w}_n$. In addition, we include atom `Is_from(i,j)` with weight $\mathtt{w}_r$ when $p(src(i,j)|\mathbf{x}) \geq \xi$, where $p(src(i,j)|\mathbf{x})$ is the probability that the relation extractor thinks opinion i and source j. In a similar fashion, we include atom `Is_about(i,j)` with weight $\mathtt{w}_r$. Also, we included priors `Implicit_src(i)` and `Implicit_trg(i)` with weights equal to the positive probability values given by the corresponding implicit prediction classifier. In contrast, we include atoms `Sense(i,s)`, `Role(i,s)` and `Overlap(i,j)` as evidences to the MLN since we directly observe them.

Second, unlike ILP, which implements the additional constraint that limits the number of arguments an opinion can take in Constraint **(3)**, MLNs cannot do so: since Markov Logic is a function-less language, MLNs in general cannot encode cardinality constraints.

Finally, as mentioned before, MLNs do not require the specification of an objective function. Instead, they infer the most likely combination of variables (MAP tuple) from the induced Markov Network obtained by grounding the MLN, typically by maximizing the conditional log-likelihood of the data.

4. MLNs: Strengths

In this section, we will discuss the strengths of MLNs.

The ability to employ soft constraints. Soft constraints are useful because (1) they allow us to model phenomena that are true with a certain probability, and (2) they provide tunable "knobs" or additional parameters that are used to increase the system's performance. A soft constraint can be created easily in a MLN by assigning a weight (i.e., a real value) to the desired constraint, where the magnitude of the weight indicates the importance that the constraint is satisfied. If it is difficult to assign weights manually, weight learning algorithms for MLNs can be used instead (e.g., Richardson and Domingos (2006), Huynh and Mooney (2011)). The ability to learn from data is one of the advantages of MLNs over ILP: unlike MLNs, ILP is a pure inference framework that does not have the ability to learn.

While it is easy to encode soft constraints in MLNs, the same is not true for ILP. One can, however, "soften" a hard constraint in ILP by incorporating it into the objective function and associating it with a coefficient. Consider, for instance, Equation (3), where the two terms in the objective function weighted using the parameter λ. Note that this parameter has a softening effect: a smaller λ implies that the first term will have less of an impact on the objective function. Hence, any hard constraint can be softened in a similar manner by associating it with a "weight" parameter when incorporated into the objective function, where the magnitude of the weight indicates the importance of satisfying the constraint. If it is difficult to manually specify the parameters, one can use a held-out development set to tune these weights.

Compact representation. While ILP operates at the propositional level, MLNs operate at the predicate level. This enables MLNs to inherit a key advantage of first-order logic over propositional logic: MLNs can encode a problem more compactly than ILP. This is appealing from a user's perspective: a user can focus on encoding the constraints on the outputs, and leave the grounding of these predicates and formulas entirely to the MLN inference engine. In particular, when applying an MLN to different problem *instances*, the first-order formulas remain unchanged: all we need to change are the evidence predicates (i.e., the observations).[5] In contrast, an ILP program cannot encode a problem instance compactly. The reason, as mentioned above, is that ILP operates at the propositional level: one variable has to be created for each decision that is to be made by the ILP

[5] A problem instance corresponds to a joint inference problem. In our fine-grained opinion extraction task, since we perform joint inference over each sentence, a problem instance corresponds to the joint inference problem created for a sentence.

inference engine, and constraints have to be defined over these variables. Given that both the objective function and the linear constraints are dependent on the variables, a different objective function and a different set of linear constraints have to be created for each problem instance.

Ease of specification. It is typically much easier to formulate constraints in MLNs than in ILP. One reason is that it is easier for humans to reason with constraints expressed as logical formulas than constraints expressed as linear equalities/inequalities. As an example, consider Constraint **(5)**, which states that an opinion candidate that is not argument-implicit must an opinion. As can be seen from Table 1, this constraint can be easily encoded as two first-order formulas, which state that an opinion candidate that is not source-implicit or target-implicit must be an opinion. On the other hand, it is comparatively more difficult for a human to *encode* and *understand* the corresponding ILP constraint: if $v_{ik} = 0$ (opinion candidate i is not implicit), then in order for this inequality to be true, x_{iO} has to be equal to 1 (i.e., i has to be an opinion).

If the above example is not compelling enough, consider encoding the transitivity constraint. If relation R is transitive, we can encode transitivity in MLN using the formula $R(i,j) \wedge R(j,k) \Rightarrow R(i,k)$. In ILP, one can enforce transitivity over three variables using the linear inequality $(1?x_{i,j}) + (1?x_{j,k}) \geq (1?x_{i,k})$. Again, it is comparatively much harder to encode and understand this constraint in ILP than in MLN. In fact, the encoding process will only become harder as a constraint involves more variables. In other words, ILP puts a much larger burden on the user than MLNs in terms of problem encoding, especially when the problem involves encoding complex output constraints.

A related point deserves mention: not all constraints that are intuitively true for a problem will indeed improve performance when they are enforced. For instance, while all the constraints in Table 1 are intuitively useful to have, not all of them will be as useful as we think: their usefulness depends on the correctness of the underlying models (i.e., the entity extractor and the relation classifiers). If these models are "reasonably accurate", enforcing the constraints could improve performance. Otherwise, enforcing the constraints could have an adverse effect on overall performance. One way to determine the usefulness of constraints is to experiment with them on a held-out development set. MLNs, however, provide an alternative solution. By encoding constraints as soft formulas, the user does not need to worry about whether employing a constraint will hurt performance: if the soft constraint is determined to be not useful, it will be assigned a low weight by the MLN.

5. MLNs: Challenges and Weaknesses

In this section, we will discuss the weaknesses of MLNs and the challenges faced by researchers when applying them to NLP tasks.

Exponential time and space complexity. Since grounding a MLN is exponential in the size of the domain and inference is shown to be NP-hard (Richardson and Domingos, 2006), handling large problems has traditionally been one of its major challenges. Fortunately, recent advances in *lifted* inference algorithms have been developed to reduce the domain size of the variables in MLNs (Gogate and Domingos, 2010), and scalable approaches to weight learning have been developed that leverage fast, approximate counting techniques (Gogate et al., 2010). Despite these advances, scalability is still a major issue surrounding inference and parameter estimation in MLNs. As of today, ILP can solve larger problems than MLNs. This in part explains the broader use of ILP in the NLP community than MLNs for joint inference.

Failure to exploit prior information in learning. As mentioned before, the confidence values assigned by the different models (e.g., the entity extractor and the relation classifiers in the fine-grained extraction task) can be incorporated as prior information for an MLN. However, existing weight learning algorithms cannot take advantage of such prior information during the training process (Venugopal et al., 2014). In particular, prior information can only be applied during test time. Moreover, since the weights were learned without taking into account the prior information, they could be suboptimal when combined with prior information to make decisions during the inference process. Typically, the priors are *scaled* before they are used in combination with the learned weights during test time (Venugopal et al., 2014).

In ILP, prior information is used within the objective function. Since there is no learning in ILP, the issue of combining priors with weight learning is not applicable to ILP.

No support for functions and preprocessing overhead. Markov Logic is a functionless language. Common operators such as equality checking and comparison are not readily available. To perform these operations, new predicates have to be defined, which may result in an expensive preprocessing step before inference and weight training. In our running example, if we were to assert that two entity candidates refer to distinct entities in Formula `4.1`, we cannot simply include the expression $i \neq j$ in it. Instead, we need to include a predicate `Neq(i,j)`, converting the formula to: `Is_from(i,j)` $\wedge$ `Neq(i,j)` $\Rightarrow$ `Span(j,S)`.[6]. This is problematic for two reasons. First, the number of additional atoms that need to be added to the network can be large depending on the domain size of the predicate's arguments. This could significantly increase the network space requirements as well as the time for inference and parameter learning. Second, the preprocessing step can be computationally expensive because the number of groundings is exponential in the size of the predicate's domain. ILP constraints, in contrast, are expressed as mathematical equations, so ILP natively support functions and therefore does not incur this preprocessing overhead. It is worth mentioning that some MLNs implementations (e.g., Rockit (Noessner et al., 2013) and Markov the Beast (Riedel, 2008)) create an ILP program as an inference subroutine that allows the use of a predefined set of functions.

[6] We circumvented this problem by including the two hard formulas in the last row of Table 1, which together have the same effect as `Neq(i,j)`.

	Overlap									Exact								
	Opinion			Target			Source			Opinion			Target			Source		
Experiment	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
ILP	50.5	72.2	**59.4**	44.9	36.2	40.1	67.3	37.4	48.1	39.2	56.1	**46.2**	15.1	12.2	13.5	57.7	32.0	41.2
MLN	75.6	45.5	56.8	55.0	34.7	**42.6**	77.1	49.6	**60.4**	59.2	35.6	44.5	25.3	16.0	**19.6**	67.7	43.6	53.0
MLN^{SRL}	66.4	47.0	55.0	51.2	25.6	34.2	64.5	59.6	62.0	52.1	36.9	43.2	15.9	8.0	10.6	55.4	51.3	**53.3**

Table 2: Entity extraction results with respect to the *overlap* and *exact* metrics.

Failure to encode cardinality constraints. Another consequence of defining Markov Logic as a functionless language is that MLNs cannot encode cardinality constraints. So, the additional cardinality constraint encoded as part of the implementation of Constraints **(3)** and **(4)** for ILP (e.g., an opinion can be related to at most three sources/targets) cannot not be enforced in an MLN. Note that some of the aforementioned MLN implementations such as *Markov the Beast* and *Rockit* are able to address this weakness. The reason is that they implement MLNs by casting the optimization problem of counting the number of satisfied constraints as an ILP problem. In other words, once a MLN is mapped to ILP, one can include cardinality constraints as in any other ILP formulations.

6. Evaluation

In this section, we will conduct an empirical comparison of our ILP and MLN for fine-grained opinion extraction.

6.1. Experimental Setup

Corpus. As mentioned in Section 2.1.2, for evaluation we use the 433 documents in the MPQA 2.0 corpus that remain after discarding those that are ill-formed. Unlike Y&C, we do not remove from these documents the sentences that contain no opinionated entities in our evaluation. Hence, our evaluation setting is arguably more challenging than that of Y&C. 20% of the corpus is reserved solely for parameter tuning (i.e., the regularization parameters in the entity extractor and the relation extraction classifiers, λ (for ILP), γ, ξ, w_e and w_r (for the MLN)). Evaluation results are obtained via five-fold cross-validation experiments on the remaining documents.

Software packages. We employ Gurobi for ILP-based joint inference and Tuffy for MLN-based joint inference.

Evaluation metrics. We use the same evaluation criteria as Y&C: precision, recall and F1 score for both *overlap* and *exact* matching mechanisms[7]. In addition, we report the time taken by the ILP/MLN package to produce the test results in each experiment.

6.2. Results and Discussion

Entity extraction results. Entity extraction results obtained using the overlap and exact metrics are shown in Table 2. Row 1 shows the ILP results. Rows 2 and 3 show the MLN results obtained without and with Constraints **(6)** and **(7)** in Table 1 (i.e., the soft constraints on the semantic role labels), respectively. Comparing rows 1 and 2, we can see that with respect to both metrics, ILP outperforms MLN on Opinion extraction but underperforms it on Source and Target extraction. The addition of the constraints on semantic roles to the MLN does not always improve its performance: comparing rows 2 and 3, we can see that with respect to both metrics, MLN^{SRL} underperforms MLN on both Opinion and Target extraction and outperforms it slightly on Source extraction. Hence, as far as entity extraction is concerned, neither inference frameworks are superior to the other, and moreover, a richer MLN model does not always yield better performance. It is worth noting that the ILP results are lower than those reported in Y&C's paper. These results suggest that retaining sentences without opinionated entities yields a harder task.[8]

	Is from			*Is about*		
Experiment	P	R	F1	P	R	F1
ILP	68.3	11.6	19.8	54.9	14.3	22.7
MLN	58.8	12.8	21.0	47.7	20.3	28.5
MLN^{SRL}	46.1	19.1	27.0	42.1	8.5	14.1

Table 3: Relation extraction results with respect to *overlap* metric.

Relation extraction results. Relation extraction results obtained using the overlap metric are shown in Table 3. The system configurations underlying the three rows in this table are the same as those in Table 2. As we can see, MLN outperforms both ILP and MLN^{SRL} with respect to both relations. It seems to have benefitted more from joint inference than ILP. In particular, its better performance on extracting both types of relations is partly responsible for its superior performance on Source and Target extraction. While MLN^{SRL} has similarly benefitted from joint inference (its relatively strong performance on extracting *Is from* relations has contributed in part to its strong performance on Source extraction), it has also suffered from joint inference. Specifically, its poor performance on extracting *Is about* relations has contributed in part to its poor performance on Target extraction. These results suggest that joint inference can sometimes improve performance and sometimes hurt performance. We speculate that semantic role information has been used correctly by MLN^{SRL} to model one but not both relation types, and that the weight learning process is not robust enough to weaken the influence of "incorrect" formulas. Additional experiments are needed to determine the reason.

[7] An overlap match occurs when a predicted entity span's indices overlap with those of a gold entity.

[8] We caution that our train-test partition may not be the same as Y&C's: their partition is not available to us.

Cardinality constraints. While the five constraints shown in Table 1 were enforced by both inference frameworks, recall that MLN does not impose the cardinality constraints that ILP employs in its encoding of Constraints **(3)** and **(4)**. The results in Tables 2 and 3 do not seem to suggest that the lack of cardinality constraints has had an adverse impact on MLN's performance.

Running time. ILP, MLN, and MLN^{SRL} took approximately 550 seconds, 7200 seconds, and 21600 seconds to produce test outputs, respectively. Note that the times shown here include not only inference time but also the time needed to read the input files, for instance. Hence, it is not entirely correct to conclude that MLN is 40 times slower than ILP. Nevertheless, this large time difference cannot be solely accounted for by differences in input reading time. We attribute the additional time used by the MLN to the *grounding* process. With the addition of predicates and formulas related to semantic roles, more predicates and formulas need to be grounded, hence the substantial increase in MLN^{SRL}'s runtime.

7. Conclusions

We conducted a qualitative comparison of two joint inference frameworks, ILP and MLNs, as well as an empirical comparison of the two on the fine-grained opinion extraction task. Empirically speaking, neither of them consistently produces superior results to the other, but our very rough approximation of running time provided suggestive evidence that ILP-based inference was more efficient than MLN-based inference. Qualitatively speaking, MLNs are superior to ILP in terms of their ease of specification as well as their ability to employ soft constraints and encode problem constraints in a compact manner. Their biggest drawback, however, is their inability to scale to large problems. Nevertheless, fast and scalable inference for MLNs is an active area of research that continuously produces advances towards more efficient algorithms.

In future work, we will conduct an extensive empirical comparison of these two frameworks on a range of NLP tasks that can potentially benefit from joint inference.

8. Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of this paper. This work was supported in part by NSF Grant IIS-1546538. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

9. Bibliographical References

Berkelaar, M., Eikland, K., Notebaert, P., et al. (2004). lp-solve: Open source (mixed-integer) linear programming system. *Eindhoven U. of Technology*.

Björkelund, A., Hafdell, L., and Nugues, P. (2009). Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics.

Domingos, P. and Lowd, D. (2009). Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–155.

Gogate, V. and Domingos, P. (2010). Exploiting logical structure in lifted probabilistic inference. In *Statistical Relational Artificial Intelligence*.

Gogate, V., Webb, W., and Domingos, P. (2010). Learning efficient markov networks. In *Advances in Neural Information Processing Systems*, pages 748–756.

Huynh, T. N. and Mooney, R. J. (2011). Online max-margin weight learning for markov logic networks. In *SDM*, pages 642–651.

Kok, S., Singla, P., Richardson, M., Domingos, P., Sumner, M., Poon, H., Lowd, D., and Wang, J. (2008). The alchemy system for statistical relational ai: user manual. *Department of Computer Science and Engineering, University of Washington*, page 41.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Niu, F., Ré, C., Doan, A., and Shavlik, J. (2011). Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *Proceedings of the VLDB Endowment*, 4(6):373–384.

Noessner, J., Niepert, M., and Stuckenschmidt, H. (2013). Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. *arXiv preprint arXiv:1304.4379*.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine learning*, 62(1-2):107–136.

Riedel, S. (2008). Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI '08)*, pages 468–475.

Riedel, S. (2009). Cutting plane map inference for markov logic. In *SRL 2009*.

Schuler, K. K. (2005). Verbnet: A broad-coverage, comprehensive verb lexicon.

Taskar, B. and Getoor, L. (2007). Introduction to statistical relational learning.

Venugopal, D. and Gogate, V. (2012). On lifting the gibbs sampling algorithm. In *Advances in Neural Information Processing Systems*, pages 1655–1663.

Venugopal, D., Chen, C., Gogate, V., and Ng, V. (2014). Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 831–843.

Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Wilson, T. A. (2008). *Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states*. Ph.D. thesis, University of Pittsburgh.

Yang, B. and Cardie, C. (2013). Joint inference for fine-grained opinion extraction. In *ACL (1)*, pages 1640–1649.