Jointly Identifying Predicates, Arguments and Senses using Markov Logic

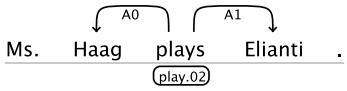
Ivan Meza-Ruiz[*] **Sebastian Riedel**[†‡]
[*]School of Informatics, University of Edinburgh, UK
[†]Department of Computer Science, University of Tokyo, Japan
[‡]Database Center for Life Science, Research Organization of Information and System, Japan
[*]I.V.Meza-Ruiz@sms.ed.ac.uk [†] sebastian.riedel@gmail.com

Abstract

In this paper we present a Markov Logic Network for Semantic Role Labelling that jointly performs predicate identification, frame disambiguation, argument identification and argument classification for all predicates in a sentence. Empirically we find that our approach is competitive: our best model would appear on par with the best entry in the CoNLL 2008 shared task open track, and at the 4th place of the closed track—right behind the systems that use significantly better parsers to generate their input features. Moreover, we observe that by fully capturing the complete SRL pipeline in a single probabilistic model we can achieve significant improvements over more isolated systems, in particular for out-of-domain data. Finally, we show that despite the joint approach, our system is still efficient.

1 Introduction

Semantic Role Labelling (SRL, Márquez et al., 2008) is generally understood as the task of identifying and classifying the semantic arguments and modifiers of the predicates mentioned in a sentence. For example, in the case of the following sentence:



we are to find out that for the predicate token "plays" with sense "play a role" (play.02) the phrase headed by the token "Haag" is referring to the player (A0) of the play event, and the phrase headed by the token "Elianti" is referring to the role (A1) being played. SRL is considered as a key task for applications that require to answer "Who", "What", "Where", etc. questions, such as Information Extraction, Question Answering and Summarization.

Any real-world SRL system needs to make several decisions, either explicitly or implicitly: which are the predicate tokens of a sentence (predicate identification), which are the tokens that have semantic roles with respect to these predicates (argument identification), which are the roles these tokens play (argument classification), and which is the sense of the predicate (sense disambiguation).

In this paper we use Markov Logic (ML), a Statistical Relational Learning framework that combines First Order Logic and Markov Networks, to develop a joint probabilistic model over all decisions mentioned above. The following paragraphs will motivate this choice.

First, it allows us to readily capture *global correlations* between decisions, such as the constraint that a predicate can only have one agent. This type of correlations has been successfully exploited in several previous SRL approaches (Toutanova et al., 2005; Punyakanok et al., 2005).

Second, we can use the joint model to evaluate the benefit of incorporating decisions into the joint model that either have not received much attention within the SRL community (predicate identification and sense disambiguation), or been largely made in isolation (argument identification and classification for all predicates of a sentence).

Third, our ML model is essentially a template that describes a class of Markov Networks. Algorithms can perform inference in terms of this template with-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 155–163,
Boulder, Colorado, June 2009.

out ever having to fully instantiate the complete Markov Network (Riedel, 2008; Singla and Domingos, 2008). This can dramatically improve the efficiency of an SRL system when compared to a propositional approach such as Integer Linear Programming (ILP).

Finally, when it comes to actually building an SRL system with ML there are "only" four things to do: preparing input data files, converting output data files, and triggering learning and inference. The remaining work can be done by an off-the-shelf Markov Logic interpreter. This is to be contrasted with pipeline systems where several components need to be trained and connected, or Integer Linear Programming approaches for which we need to write additional wrapper code to generate ILPs.

Empirically we find that our system is competitive—our best model would appear on par with the best entry in the CoNLL 2008 shared task open track, and at the 4th place of the closed track—right behind systems that use significantly better parsers[1] to generate their input features.

We also observe that by integrating frame disambiguation into the joint SRL model, and by extracting all arguments for all predicates in a sentence simultaneously, significant improvements compared to more isolated systems can be achieved. These improvements are particularly large in the case of out-of-domain data, suggesting that a joint approach helps to increase the robustness of SRL. Finally, we show that despite the joint approach, our system is still efficient.

Our paper is organised as follows: we first introduce ML (section 2), then we present our model in terms of ML (section 3) and illustrate how to perform learning and inference with it (section 4). How this model will be evaluated is explained in section 5 with the corresponding evaluation presented in section 6. We conclude in section 7.

2 Markov Logic

Markov Logic (ML, Richardson and Domingos, 2005) is a Statistical Relational Learning language based on First Order Logic and Markov Networks. It can be seen as a formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative point of view, it is an expressive template language that uses First Order Logic formulae to instantiate Markov Networks of repetitive structure.

Let us describe ML by considering the predicate identification task. In ML we can model this task by first introducing a set of logical predicates[2] such as *isPredicate(Token)* or *word(Token,Word)*. Then we specify a set of weighted first order formulae that define a distribution over sets of ground atoms of these predicates (or so-called *possible worlds*).

Ideally, the distribution we define with these weighted formulae assigns high probability to possible worlds where SRL predicates are correctly identified and a low probability to worlds where this is not the case. For example, a suitable set of weighted formulae would assign a high probability to the world[3]

$$\{word\,(1, \text{Haag})\,, word(2, \text{plays}),\\ word(3, \text{Elianti}), isPredicate(2)\}$$

and a low one to

$$\{word\,(1, \text{Haag})\,, word(2, \text{plays}),\\ word(3, \text{Elianti}), isPredicate(3)\}$$

In Markov Logic a set of weighted formulae is called a *Markov Logic Network* (MLN). Formally speaking, an MLN M is a set of pairs (ϕ, w) where ϕ is a first order formula and w a real weight. M assigns the probability

$$p\,(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{(\phi,w)\in M} w \sum_{\mathbf{c}\in C^\phi} f^\phi_{\mathbf{c}}\,(\mathbf{y})\right) \quad (1)$$

to the possible world $\mathbf{y}$. Here C^ϕ is the set of all possible bindings of the free variables in ϕ with the constants of our domain. $f^\phi_{\mathbf{c}}$ is a feature function that returns 1 if in the possible world $\mathbf{y}$ the *ground formula* we get by replacing the free variables in ϕ by the constants in $\mathbf{c}$ is true and 0 otherwise. Z is a normalisation constant. Note that this distribution corresponds to a Markov Network (the so-called *Ground Markov Network*) where nodes represent ground atoms and factors represent ground formulae.

[1] Our unlabelled accuracy for syntactic dependencies is at least 3% points under theirs.

[2] In the cases were is not obvious whether we refer to SRL or ML predicates we add the prefix SRL or ML, respectively.

[3] "Haag plays Elianti" is a segment of a sentence in the training corpus.

For example, if M contains the formula ϕ

$$word\,(x, \text{take}) \Rightarrow isPredicate\,(x)$$

then its corresponding log-linear model has, among others, a feature f^{ϕ}_{t1} for which x in ϕ has been replaced by the constant t_1 and that returns 1 if

$$word\,(1, \text{take}) \Rightarrow isPredicate\,(1)$$

is true in $\mathbf{y}$ and 0 otherwise.

We will refer predicates such as *word* as *observed* because they are known in advance. In contrast, *isPredicate* is *hidden* because we need to infer it at test time.

3 Model

Conceptually we divide our SRL system into three stages: one stage that identifies the predicates of a sentence, one stage that identifies and classifies the arguments of these predicates, and a final stage that predicts the sense of each predicate. We should stress that this architecture is intended to illustrate a typical SRL system, and to describe the pipeline-based approach we will compare our models to. However, it does *not* correspond to the way inference is performed in our proposed model—we jointly infer *all* decisions described above.

Note that while the proposed division into conceptual stages seems somewhat intuitive, it is by no means uncontroversial. In fact, for the CoNLL 2008 shared task slightly more than one half of the participants performed sense disambiguation before argument identification and classification; most other participants framed the problem in the reverse order.[4]

We define five hidden predicates for the three stages of the task. Figure 1 illustrates these predicates and the stage they belong to. For predicate identification, we use the predicate *isPredicate*. *isPredicate(p)* indicates that the word in the position p is an SRL predicate. For argument identification and classification, we use the predicates *isArgument*, *hasRole* and *role*. The atom *isArgument(a)* signals that the word in the position a is a SRL argument of some (unspecified) SRL predicate while *hasRole(p,a)* indicates that the token at position a is

[4]However, for almost all pipeline based systems, predicate identification was the first stage of the role labelling process.

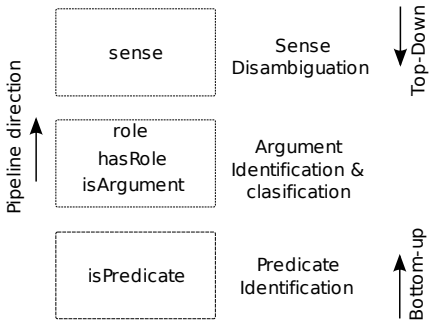


Figure 1: MLN hidden predicates divided in stages

an argument of the predicate in position p. The predicate *role(p,a,r)* corresponds to the decision that the argument at position a has the role r with respect to the predicate in position p. Finally, for sense disambiguation we define the predicate *sense(p,e)* which signals that the predicate in position p has the sense e.

Before we continue to describe the formulae of our Markov Logic Network we would like to highlight the introduction of the $isArgument$ predicate mentioned above. This predicate corresponds to a decision that is usually made implicitly: a token is an argument if there exists a predicate for which it plays a semantic role. Here we model this decision explicitly, assuming that there exist cases where a token clearly has to be an argument of some predicate, regardless of which predicate in the sentence this might be. It is this assumption that requires us to infer the arguments for all predicates of a sentence at once—otherwise we cannot make sure that for a marked argument there exists at least one predicate for which the argument plays a semantic role.

In addition to the hidden predicates, we define observable predicates to represent the information available in the corpus. Table 1 presents these predicates.

3.1 Local formulae

A formula is *local* if its groundings relate any number of observed ground atoms to exactly one hidden ground atom. For example, two groundings of the local formula

$$lemma(p, +l_1) \wedge lemma(a, +l_2) \Rightarrow hasRole(p, a)$$

can be seen in the Factor Graph of Figure 2. Both connect a single hidden *hasRole* ground atom with

word(i,w)	Token i has word w
lemma(i,l)	Token i has lemma l
ppos(i,p)	Token i has POS tag p
cpos(i,p)	Token i has coarse POS tag p
voice(i,v)	Token i is verb and has voice v (Active/Passive).
subcat(i,f)	Token i has subcategorization frame f
dep(i,j,d)	Token h is head of token m and has dependency label d
palmer(i,j)	Token j can be semantic argument for token i according to high recall heuristic*
depPath(i,j,p)	Dependency path between tokens i and j is p*
depFrame(i,j,f)	f is a syntactic (dependency) frame in which tokens i and j are designated as "pivots"*

Table 1: Observable predicates; predicates marked with * are dependency parsing-based versions for features of Xue and Palmer (2004).

two observed *lemma* ground atoms. The + notation indicates that the MLN contains one instance of the rule, with a separate weight, for each assignment of the variables with a plus sign (**?**).

The local formulae for *isPredicate*, *isArgument* and *sense* aim to capture the relation of the tokens with their lexical and syntactic surroundings. This includes formulae such as

$$subcat(p, +f) \Rightarrow isPredicate(p)$$

which implies that a certain token is a predicate with a weight that depends on the subcategorization frame of the token. Further local formulae are constructed using those observed predicates in table 1 that relate single tokens and their properties.

The local formulae for *role* and *hasRole* focus on properties of the predicate and argument token—the formula illustrated in figure 2 is an example of this—and on the relation between the two tokens. An example of the latter type is the formula

$$depPath(p, a, +d) \Rightarrow role(p, a, +r)$$

which implies that token a plays the semantic role r with respect to token p, and for which the weight depends on the syntactic (dependency) path d between p and a and on the actual role to assign. Again, further formulae are constructed using the observed

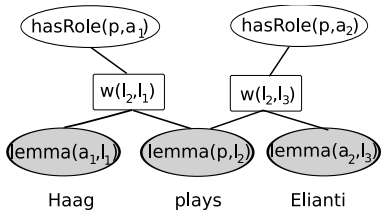


Figure 2: Factor graph for the first local formula in section 3.1. Here round nodes represent variables (corresponding to the states of ground atoms) and the rectangular nodes represent the factor and their parameters attached to the ground formulae.

predicates in table 1; however, this time we consider both predicates that relate tokens to their individual properties and predicates that describe the relation between tokens.

Unfortunately, the complete set of local formulae is too large to be exhaustively described in this paper. Its size results from the fact that we also consider conjunctions of several atoms as conditions, and lexical windows around tokens. Hence, instead of describing all local formulae we refer the reader to our MLN model files.[5] They can be used both as a reference and as input to our Markov Logic Engine,[6] and thus allow the reader to easily reproduce our results.

3.2 Global formulae

Global formulae relate several hidden ground atoms. We use this type of formula for two purposes: to ensure consistency between the predicates of all SRL stages, and to capture some of our background knowledge about SRL. We will refer to formulae that serve the first purpose as *structural constraints*.

For example, a structural constraint is given by the (deterministic) formula

$$role(p, a, r) \Rightarrow hasRole(p, a)$$

which ensures that, whenever the argument a is given a label r with respect to the predicate p, this argument must be an argument of a as denoted by *hasRole(p,a)*. Note that this formula by itself models the traditional "bottom-up" argument identification and classification pipeline (Xue and Palmer, 2004):

[5] http://code.google.com/p/thebeast/source/browse/#svn/mlns/naacl-hlt

[6] http://code.google.com/p/thebeast

it is possible to not assign a role r to an predicate-argument pair (p, a) proposed by the identification stage; however, it is impossible to assign a role r to token pairs (p, a) that have not been proposed as potential arguments.

An example of another class of structural constraints is

$$hasRole(p, a) \Rightarrow \exists r.role(p, a, r)$$

which, by itself, models an inverted or "top-down" pipeline. In this architecture the argument classification stage can assign roles to tokens that have not been proposed by the argument identification stage. However, it must assign a label to any token pair the previous stage proposes.

For the SRL predicates that perform a labelling task (*role* and *sense*) we also need a structural constraint which ensures that not more than one label is assigned. For instance,

$$(role(p, a, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(p, a, r_2))$$

forbids two different semantic roles for a pair of words.

There are three global formulae that capture our linguistic background knowledge. The first one is a deterministic constraint that had been frequently applied in the SRL literature. It forbids cases where distinct arguments of a predicate have the same role unless the role describes a modifier:

$$role\,(p, a_1, r) \wedge \neg mod\,(r) \wedge a_1 \neq a_2 \Rightarrow \neg role\,(p, a_2, r)$$

The second "linguistic" global formula is

$$role(p, a, +r) \wedge lemma(p, +l) \Rightarrow sense(p, +s)$$

which implies that when a predicate p with lemma l has an argument a with role r it has to have the sense s. Here the weight depends on the combination of role r, lemma l and sense s.

The third and final "linguistic" global formula is

$$lemma(p, +l) \wedge ppos(a, +p) \wedge hasRole(p, a) \Rightarrow sense(p, +f)$$

It implies that if a predicate p has the lemma l and an argument a with POS tag p it has to have the sense s. This time the weight depends on the combination of POS tag p, lemma l and sense s.

Note that the final two formulae evaluate the semantic frame of a predicate and become local formulae in a pipeline system that performs sense disambiguation after argument identification and classification.

Table 2 summarises the global formulae we use in this work.

4 Inference and Learning

Assuming that we have an MLN, a set of weights and a given sentence then we need to predict the choice of predicates, frame types, arguments and role labels with maximal *a posteriori* probability (MAP). To this end we apply a method that is both exact and efficient: Cutting Plane Inference (CPI, Riedel, 2008) with Integer Linear Programming (ILP) as *base solver*.

Instead of fully instantiating the Markov Network that a Markov Logic Network describes, CPI begins with a subset of factors/edges—in our case we use the factors that correspond to the local formulae of our model—and solves the MAP problem for this subset using the base solver. It then inspects the solution for ground formulae/features that are not yet included but could, if added, lead to a different solution—this process is usually referred to as *separation*. The ground formulae that we have found are added and the network is solved again. This process is repeated until the network does not change anymore.

This type of algorithm could also be realised for an ILP formulation of SRL. However, it would require us to write a dedicated separation routine for each type of constraint we want to add. In Markov Logic, on the other hand, separation can be generically implemented as the search for variable bindings that render a weighted first order formulae true (if its weight is negative) or false (if its weight is positive). In practise this means that we can try new global formulae/constraints without any additional implementation overhead.

We learn the weights associated with each MLN using 1-best MIRA (Crammer and Singer, 2003) Online Learning method. As MAP inference method that is applied in the inner loop of the online learner we apply CPI, again with ILP as base

Bottom-up	$sense(p, s) \Rightarrow isPredicate(p)$ $hasRole(p, a) \Rightarrow isPredicate(p)$ $hasRole(p, a) \Rightarrow isArgument(a)$ $role(p, a, r) \Rightarrow hasLabel(p, a)$
Top-Down	$isPredicate(p) \Rightarrow \exists s.sense(p, s)$ $isPredicate(p) \Rightarrow \exists a.hasRole(p, a)$ $isArgument(a) \Rightarrow \exists p.hasRole(p, a)$ $hasLabel(p, a) \Rightarrow \exists r.role(p, a, r)$
Unique Labels	$role(p, a, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(p, a, r_2)$ $sense(p, s_1) \wedge s_1 \neq s_2 \Rightarrow \neg sense(p, r_2)$
Linguistic	$role\,(p, a_1, r) \wedge \neg mod\,(r) \wedge a_1 \neq a_2 \Rightarrow \neg role\,(p, a_2, r)$ $lemma(p, +l) \wedge ppos(a, +p) \wedge hasRole(p, a) \Rightarrow sense(p, +f)$ $lemma(p, +l) \wedge role(p, a, +r) \Rightarrow sense(p, +f)$

Table 2: Global formulae for ML model

solver.

5 Experimental Setup

For training and testing our SRL systems we used a version of the CoNLL 2008 shared task (Surdeanu et al., 2008) dataset that only mentions verbal predicates, disregarding the nominal predicates available in the original corpus.[7] While the original (open track) corpus came with MALT (Nivre et al., 2007) dependencies, we observed slightly better results when using the dependency parses generated with a Charniak parser (Charniak, 2000). Hence we used the latter for all our experiments.

To assess the performance of our model, and it to evaluate the possible gains to be made from considering a joint model of the complete SRL pipeline, we set up several systems. The *full* system uses a Markov Logic Network with all local and global formulae described in section 3. For the *bottom-up* system we removed the structural top-down constraints from the complete model—previous work Riedel and Meza-Ruiz (2008) has shown that this can lead to improved performance. The *bottom-up (-arg)* system is equivalent to the bottom-up system, but it does not include any formulae that mention the hidden $isArgument$ predicate.

For the systems presented so far we perform joint inference and learning. The *pipeline* system differs in this regard. For this system we train a separate model for each stage in the pipeline of figure 1. The predicate identification stage identifies the predicates (using all local $isPredicate$ formulae) of a sentence. The next stage predicts arguments and their roles for the identified predicates. Here we include all local and global formulae that involve only the predicates of this stage. In the last stage we predict the sense of each identified predicate using all formulae that involve the $sense$, without the structural constraints that connect the $sense$ predicate to the previous stages of the pipeline (these constraints are enforced by architecture).

6 Results

Table 3 shows the results of our systems for the CoNLL 2008 development set and the WSJ and brown test sets. The scores are calculated using the semantic evaluation metric of the CoNLL-08 shared task (Surdeanu et al., 2008). This metric measures the precision, recall and F_1 score of the recovered semantic dependencies. A semantic dependency is created for each predicate and its arguments, the label of such dependency is the role of the argument. Additionally, there is a semantic dependency for each predicate and a $ROOT$ argument which has the sense of the predicate as label.

To put these results into context, let us compare them to those of the participants of the CoNLL 2008 shared task (see the last three rows of table 3).[8] Our best model, Bottom-up, would reach the highest F_1 WSJ score, and second highest Brown score, for the open track. Here the best-performing participant was Vickrey and Koller (2008).

Table 3 also shows the results of the best (Johansson and Nugues, 2008) and fourth best sys-

[7] The reason for this choice where license problems.

[8] Results of other systems were extracted from Table 16 of the shared task overview paper (Surdeanu et al., 2008).

tem (Zhao and Kit, 2008) of the closed track. We note that we do significantly worse than Johansson and Nugues (2008), and roughly equivalent to Zhao and Kit (2008); this places us on the fourth rank of 19 participants. However, note that all three systems above us, as well as Zhao and Kit (2008), use parsers with at least about 90% (unlabelled) accuracy on the WSJ test set (Johansson's parser has about 92% unlabelled accuracy).[9] By contrast, with about 87% unlabelled accuracy our parses are significantly worse.

Finally, akin to Riedel and Meza-Ruiz (2008) we observe that the bottom-up joint model performs better than the full joint model.

System	Devel	WSJ	Brown
Full	76.93	79.09	67.64
Bottom-up	77.96	80.16	68.02
Bottom-up (-arg)	77.57	79.37	66.70
Pipeline	75.69	78.19	64.66
Vickrey	N/A	79.75	69.57
Johansson	N/A	86.37	71.87
Zhao	N/A	79.40	66.38

Table 3: Semantic F_1 scores for our systems and three CoNLL 2008 shared task participants. The Bottom-up results are statistically significantly different to all others (i.e., $\rho \leq 0.05$ according to the sign test).

6.1 Joint Model vs. Pipeline

Table 3 suggests that by including sense disambiguation into the joint model (as is the case for all systems but the pipeline) significant improvements can be gained. Where do these improvements come from? We tried to answer this question by taking a closer look at how accurately the pipeline predicts the $isPredicate$, $isArgument$, $hasRole$, $role$ and $sense$ relations, and how this compares to the result of the joint full model.

Table 4 shows that the joint model mainly does better when it comes to predicting the right predicate senses. This is particularly true for the case of the Brown corpus—here we gain about 10% points. These results suggest that a more joint approach may be particularly useful in order to increase the robustness of an SRL system in out-of-domain scenarios.[10]

[9] Since our parses use a different label set we could not compare labelled accuracy.

	WSJ		Brown	
	Pipe.	Fu.	Pipe.	Fu.
isPredicate	96.6	96.5	92.2	92.5
isArgument	90.3	90.6	85.9	86.9
hasRole	88.0	87.9	83.6	83.8
role	75.4	75.5	64.2	64.6
sense	85.5	88.5	67.3	77.1

Table 4: F_1 scores for M predicates; Pipe. refers to the Pipeline system, Fu. to the full system.

6.2 Modelling if a Token is an Argument

In table 3 we also observe that improvements can be made if we explicitly model the decision whether a token is a semantic argument of some predicate or not. As we mentioned in section 3, this aspect of our model requires us to jointly perform inference for all predicates of a sentence, and hence our results justify the per-sentence SRL approach proposed in this paper.

In order to analyse where these improvements come from, we again list our results on a per-SRL-predicate basis. Table 5 shows that by including the $isArgument$ predicate and the corresponding formulae we gain around 0.6% and 1.0% points across the board for WSJ and Brown, respectively.[11] As shown in table 3, these improvements result in about 1.0% improvements for both WSJ and Brown in terms of the CoNLL 2008 metric. Hence, an explicit model of the "is an argument" decision helps the SRL at all levels.

How the $isArgument$ helps to improve the overall role labelling score can be illustrated with the example in figure 3. Here the model without a hidden $isArgument$ predicate fails to attach the preposition "on" to the predicate "start.01" (here 01 refers to the sense of the predicate). Apparently the model has not enough confidence to assign the preposition to either "start.01" or "get.03", so it just drops the argument altogether. However, because the $isArgument$ model knows that most prepositions have to be modifying some predicate, pres-

[10] The differences between results of the full and joint model are statistically significant with the exception of the results for the $isPredicate$ predicate for the WSJ test set.

[11] The differences between results of the w/ and w/o model are statistically significant with the exception of the results for the $sense$ predicate for the Brown test set.

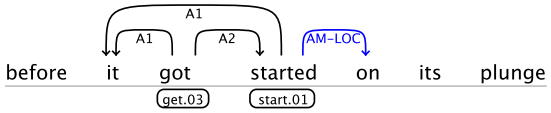


Figure 3: Segment of the CoNLL 2008 development set for which the bottom-up model w/o $isArgument$ predicate fails to attach the preposition "on" as an "AM-LOC" for "started". The joint bottom-up model attaches the preposition correctly.

sure is created that forces a decision between the two predicates. And because for the role model "start.01" looks like a better fit than "get.03", the correct attachment is found.

	WSJ		Brown	
	w/o	w/	w/o	w/
isPredicate	96.3	96.5	91.4	92.5
hasRole	87.1	87.7	82.5	83.6
role	76.9	77.5	65.2	66.2
sense	88.3	89.0	76.1	77.5

Table 5: F_1 scores for ML predicates; w/o refers to a Bottom-up system without isArgument predicate, w/ refers to a Bottom-up system with isArgument predicate.

6.3 Efficiency

In the previous sections we have shown that our joint model indeed does better than an equivalent pipeline system. However, usually most joint approaches come at a price: efficiency. Interestingly, in our case we observe the opposite: our joint model is actually faster than the pipeline. This can be seen in table 6, where we list the time it took for several different system to process the WSJ and Brown test corpus, respectively. When we compare the times for the bottom-up model to those of the pipeline, we note that the joint model is twice as fast. While the individual stages within the pipeline may be faster than the joint system (even when we sum up inference times), extracting results from one system and feeding them into another creates overhead which offsets this potential reduction.

Table 6 also lists the run-time of a bottom-up system that solves the inference problem by fully grounding the Markov Network that the Markov Logic (ML) model describes, mapping this network to an Integer Linear Program, and finding the most likely assignment using an ILP solver. This system (Bottom-up (-CPI)) is four times slower than the equivalent system that uses Cutting Plane Inference (Bottom-up). This suggests that if we were to implement the same joint model using ILP instead of ML, our system would either be significantly slower, or we would need to implement a Cutting Plane algorithm for the corresponding ILP formulation—when we use ML this algorithm comes "for free".

System	WSJ	Brown
Full	9.2m	1.5m
Full (-CPI)	38.4m	7.47m
Bottom-up	9.5m	1.6m
Bottom-up (-CPI)	38.8m	6.9m
Pipeline	18.9m	2.9m

Table 6: Testing times for full model and bottom-up when CPI algorithm is not used. The WSJ test set contains 2414 sentences, the Brown test set 426. Our best systems thus takes on average 230ms per WSJ sentence (on a 2.4Ghz system).

7 Conclusion

In this paper we have presented a Markov Logic Network that jointly models all predicate identification, argument identification and classification and sense disambiguation decisions for a sentence. We have shown that this approach is competitive, in particular if we consider that our input parses are significantly worse than those of the top CoNLL 2008 systems.

We demonstrated the benefit of jointly predicting senses and semantic arguments when compared to a pipeline system that first picks arguments and then senses. We also showed that by modelling whether a token is an argument of some predicate and jointly picking arguments for all predicates of a sentence, further improvements can be achieved.

Finally, we demonstrated that our system is efficient, despite following a global approach. This efficiency was also shown to stem from the first order inference method our Markov Logic engine applies.

Acknowledgements

The authors are grateful to Mihai Surdeanu for providing the version of the corpus used in this work.

References

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*, 2000.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

Richard Johansson and Pierre Nugues. Dependency-based semantic role labeling of propbank. In *Proceedings of EMNLP-2008.*, 2008.

Lluís Márquez, Xavier Carreras, Ken Litkowski, and Suzanne Stevenson. Semantic role labeling. *Computational Linguistics*, 34(2), 2008. Introduction to the Special Issue on Semantic Role Labeling.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kuebler, S. Marinov, and E. Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007.

V. Punyakanok, D. Roth, and W. Yih. Generalized inference with multiple semantic role labeling systems. In Ido Dagan and Dan Gildea, editors, *CoNLL '05: Proceedings of the Annual Conference on Computational Natural Language Learning*, pages 181–184, 2005.

Matthew Richardson and Pedro Domingos. Markov logic networks. Technical report, University of Washington, 2005.

Sebastian Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *UAI '08: Proceedings of the Annual Conference on Uncertainty in AI*, 2008.

Sebastian Riedel and Ivan Meza-Ruiz. Collective semantic role labelling with markov logic. In *Conference on Computational Natural Language Learning*, 2008.

P. Singla and P. Domingos. Lifted First-Order Belief Propagation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2008.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, 2008.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. Joint learning improves semantic role labeling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Morristown, NJ, USA, 2005.

David Vickrey and Daphne Koller. Applying sentence simplification to the conll-2008 shared task. In *Proceedings of CoNLL-2008.*, 2008.

Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *EMNLP '04: Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing*, 2004.

Hai Zhao and Chunyu Kit. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, Manchester, England, 2008.