Machine Reading Using Markov Logic Networks for Collective Probabilistic Inference

Shalini Ghosh, Natarajan Shankar, Sam Owre

Computer Sciences Laboratory, SRI International

Abstract. DARPA's *Machine Reading* project is directed at extracting specific information from natural language text such as events from news articles. We describe a component of FAUST, a system designed for machine reading, which combines state-of-the-art information extraction (IE), based on statistical parsing and local sentence-wise analysis, with global article-wide inference using Markov Logic Networks (MLNs). We use the probabilistic first-order rules of MLNs to encode domain knowledge for resolving ambiguities and inconsistencies of the extracted information, and for merging multiple information sources. Experiments over a corpus of NFL news articles show that collective inference using the probabilistic relational MLN engine substantially improves the performance, according to F-measure, over individual IE engines.

1 Introduction

A lot of information is embedded in natural language text in books, newspaper and magazine articles, and, increasingly, in web pages, wikis, and blogs. If this information can be extracted and represented in a database system, then it can be retrieved and analyzed in useful ways. This is the larger goal of DARPA's Machine Reading project. Statistical techniques for natural language parsing have already reached a practical level of proficiency. They can parse sentences to return the most likely or k best parse structures. Information can then be extracted from the parse structures to identify the entities and relations that are mentioned in a sentence. Information extracted purely from syntactic considerations might contain errors and ambiguities, which can be resolved using domain knowledge and information from other parts of the document. We describe the use of probabilistic relational models for performing collective inference at the document-level, to improve the accuracy of information extracted at the sentence-level. Note that the information extraction system does some inference across multiple sentences, e.g., identifying that the same game is being referred to across different sentences. We represent domain knowledge in the form of probabilistic first-order logic rules given by MLNs [17]. The MLN-based reasoning system associates weighted semantic rules (derived from domain knowledge) with syntactic information (extracted by the IE system) for doing collective probabilistic inference across the document. This probabilistic inference component is part of a larger system for machine reading, called FAUST – Flexible Acquisition and Understanding System for Text (see Figure 1).

Consider this passage from a sports article: *With Favre throwing the best-looking pinpoint passes this side of Troy Aikman and with receiver Robert Brooks doing a great impression of Michael Irvin and with the Packers' defense playing like, well, like themselves, GreenBay routed Philadelphia, 39-13. These days, the Packers are looking a lot like the Cowboys. Favre, who went out early in the fourth quarter with a 39-7 lead, completed 17 of 31 passes for 261 yards and three touchdowns. He has thrown seven touchdown passes in his first two games with no interceptions. In the season opener*

against the Buccaneers last week, he picked up where he left off in 1995. He threw four touchdown passes and guided the Packers to a 34-3 victory over Tampa Bay.

We can assume that the article is about a specific game and that we have a background ontology that identifies critical entities and relations in the National Football League (NFL) such as teams, players, coaches, touchdowns, passes, interceptions, and quarters. A knowledgeable reader could, with some confidence, infer from the text that

1. The game in question is between GreenBay Packers and Philadelphia Eagles.
2. GreenBay Packers won this game with a final score of 39 to 13.
3. In this game, Brett Favre's team, the GreenBay Packers, had a 39 to 7 lead early in the fourth quarter.
4. Brett Favre completed 17 of 31 passes.
5. He had no interceptions in the first two games.
6. The GreenBay Packers played the Tampa Bay Buccaneers in the first game.
7. This game was won by the GreenBay Packers with a score of 34 to 3.
8. The score of 34 to 3 is a final score, but not for the game in question.

Extracting this kind of information requires reasoning from domain knowledge, some specifically about Football. For example, we need to know that the final score is never smaller than an intermediate score since points can only be given but never taken away. We can also be reasonably certain that the "39 to 7" lead refers to the current game, since the score is higher than the final score of the first game. However, we can already achieve a fair degree of disambiguation from syntactic considerations. For example, we can conclude that 261 refers to yardage, "17 of 31" is not a score, and that 1995 is a year. Similarly, we can, from syntactic considerations, conclude that the anaphora "He" in the fourth sentence refers to Brett Favre since it follows the sentence where his name appears as a proper noun. However, semantic knowledge can help strengthen this conviction using the knowledge that Brett Favre is the quarterback of the GreenBay Packers and it is the quarterback who typically throws passes. Similarly, in the sentences below, semantic considerations can be used to guess that "Chicago" refers to the football team in the first sentence and to the city in the second sentence.

1. Chicago enjoyed a home-field advantage.

2. Chicago enjoyed a spell of warm weather.

In this way, semantic constraints can be used to sharpen the information extracted by syntactic means in order to disambiguate references and co-references across the entire document. The challenge then is to develop semantic information extraction methods that can inter-operate with syntactic ones. In particular, semantic information extraction should improve the accuracy of syntactic information extraction from one or more sources. Our approach to semantic information extraction is based on collective document-level probabilistic inference with MLNs. An MLN is a collection of weighted and unweighted first-order formulas that capture specific rules. An unweighted rule is a hard rule that holds in every model. For example, an unweighted rule might say that a touchdown scored by a player is scored for the team. On the other hand, a rule that says that passes are thrown by the quarterback has a high weight attached to indicate that it has a small probability of being false.

FAUST, our system for machine reading, extracts relevant information for automated reasoning using a tight integration between NLP modules and a powerful probabilistic inference engine. In FAUST, information is extracted in the syntactic phase in the form of logical assertions (facts) with associated probabilities. Semantic information is then extracted from

the system by applying probabilistic inference to the syntactic information, using an MLN capturing the domain-specific rules. We use SRI's Probabilistic Consistency Engine (PCE) for probabilistic inference in MLNs [14]. With PCE, a knowledge base *KB* consisting of weighted assertions and rules can be queried for the probability associated with a formula F, e.g., in the NFL domain, the queries include game winner or the game loser. Modeling domain-specific rules with subjective probabilities in the form of an MLN can be quite complex, as outlined in Section 3.

The main contribution of our paper is an approach to augmenting local syntactic information extraction with collective inference using probabilistic first-order semantic rules. We have designed a notation – the Common Annotation Format (CAF) – for capturing syntactic information, and developed an approach for expressing domain knowledge in the form of MLN rules. We conducted experiments applying probabilistic inference to the CAF output of the IE engine with the MLN rules for evaluating specific queries on the natural language text. The experimental results show that using probabilistic inference substantially improves performance (according to F-measure) over the IE engine for predicting the game winner in a corpus of 500 NFL new articles (details in Section 4). Section 2 gives some background in probabilistic inference, as implemented in PCE. Section 3 describes the use of PCE in the Machine Reading project. Section 4 presents an experimental evaluation of the performance of PCE on a specific natural language corpus. A discussion of related work is given in Section 5, and the paper ends with conclusion and discussion of future work in Section 6.

2 Background

We give some background for the relevant parts of the FAUST system – in particular, we discuss the IE component of the NLP modules (specifically Stanford's NLP engine), and the Probabilistic Consistency Engine (PCE) component of the probabilistic reasoning system.

2.1 Information Extraction Systems

The IE engines in the NLP modules use state-of-the-art statistical NLP, parsing and disambiguation to extract entities and relations from each sentence in the text document. FAUST uses multiple IE systems to extract entities/relations from the text. To make it possible for PCE to handle the output of multiple IE systems, we have designed CAF, a common format into which the output of each IE engine is converted. An entity/relation encoded in CAF is essentially a set of key-value pairs encoding the entity/relation, specified in the JSON format. [1] The CAF format keeps track of various IDs (e.g., DocumentID, SentenceID, EntityMentionID, RelationMentionID), which are used in FAUST for cross-referencing different occurrences of sentences, references for anaphora/co-reference resolution, entities, relations, etc. between the IE engine and the down-stream probabilistic inference module. We also need to normalize certain entity values. For example, the team "GreenBayPackers" is referred to in the text in different forms, (e.g., "GreenBay" or "Packers") – these references are normalized to the canonical name "GreenBayPackers" to enable probabilistic reasoning with this entity. The CAF format facilitates such a canonical representation. The overall system architecture of FAUST is given in Figure 1.

[1] `http://www.hunlock.com/blogs/Mastering_JSON_(_JavaScript_Object_Notation_)`

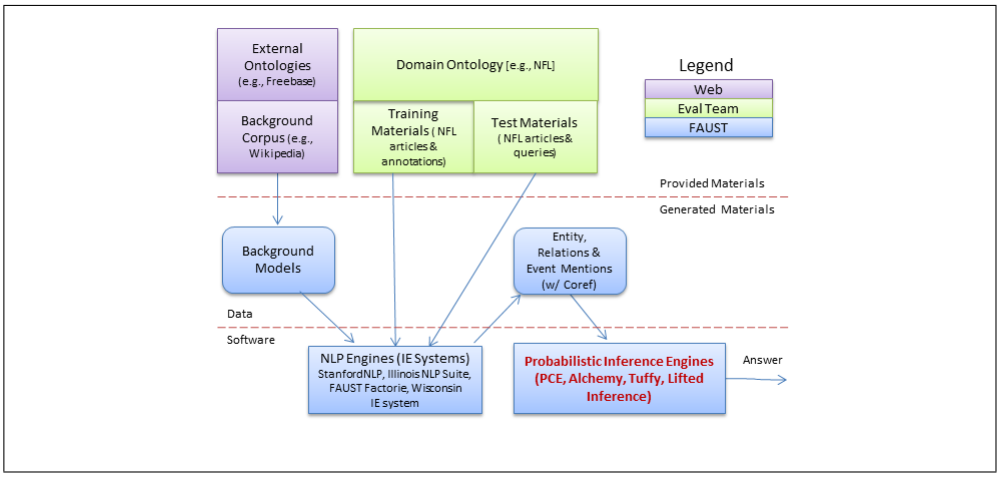

Fig. 1: Role of PCE in FAUST (modified from original figure by Lynn Voss).

Tasks like parsing, co-reference resolution and generic entity finding are being done by the Stanford CoreNLP system [21],[2] followed by processing by an additional domain-specific entity and relation extraction system or IE system (also provided by Stanford), augmented with a set of basic inference rules for consistency checking. We henceforth refer to the Stanford CoreNLP system + IE system + Consistency Checking rules as the StanfordNLP system, for ease of reference. In Section 4, we compare the performance of StanfordNLP with PCE.

2.2 Probabilistic Consistency Engine (PCE)

SRI's PCE is a tool that does efficient inference with probabilistic first-order rules, using the framework of MLNs.

Markov Logic Networks: An MLN [17] is a statistical relational model to formalize first-order logic with probabilities. In an MLN, all random variables are Boolean and all feature functions are Boolean formulas. The formulas in the MLN have weights that are associated with their probabilities – given a knowledge base (conjunction of formulas), the weights on the formulas are used to compute the associated model probability in the KB. One can then compute the marginal probability of a given formula F as the probability aggregate over the models where F evaluates to true. In typical use, an MLN is used to infer the marginal probability distributions for (output) random variables and formulas based on the distribution for the input random variables, over the space of models defined by the formulas and their corresponding probabilities. The MC-SAT inference algorithm computes these marginal probabilities by efficiently averaging the probabilities over a sequence of models.

PCE: PCE uses multi-sorted first-order logic for describing a network of formulas and weights to build probabilistic relational models. PCE provides a language for representing MLNs and implements an optimized version of the MC-SAT algorithm of Poon and Domingos for probabilistic inference [16], to compute marginal probabilities of formulas. PCE uses a combination of simulated annealing/SampleSAT and WalkSAT to implement MCMC

[2] `http://nlp.stanford.edu/software/corenlp.shtml`

sampling for estimating marginal probabilities – WalkSAT is used to build the initial model, where SampleSAT/simulated annealing are used in each subsequent iteration.

The object language of PCE consists of sorts, literals, constants, and formulas. Type information is encoded by the sorts, rules are represented as universally quantified formulas in both the observable and hidden predicates along with their associated weights, while facts are represented as grounded instances of literals and formulas. Various probabilistic inference problems can be represented as MLNs in the form of facts and weighted and unweighted rules. The weight of a model is given by the weight of the formulas that hold in the model, and the probability of a formula is the normalized sum of the weights of the models in which the formula holds. PCE outputs the marginal probabilities for the atomic formulas and any query formulas. PCE is order-sorted since it also has subsorts.

PCE models in FAUST: As mentioned earlier, the extracted facts (relations and entities with associated probabilities) are sent to PCE for probabilistic reasoning. This is done through building MLN models in PCE. Figure 2 shows such an MLN – the jointWinnerLoser MLN, which is used in FAUST for answering the queries related to the gameWinner and gameLoser from an NFL document. The "Definitions" block defines the sorts and the predicates that are used in this MLN. The following block of "Rules" has the main domain-specific rules of the MLN, which are segmented into rules used for inferring different predicates, e.g., rules for teamInGame, rules for finalScored. At the end of the "Rules" block, we have the block of "Facts" from the output of StanfordNLP, which are grounded instances of entities and relations in the MLN. The final "Query" block specifies the parameters used for MC-SAT and the actual queries to the MLN to obtain the gameWinner and gameLoser predictions.

One thing to note here is that the relations obtained from the IE (e.g., teamScoringAll, teamInGame) engine and the relations in the MLN (e.g., gameWinner, gameLoser) are defined in an NFL ontology in the FAUST system. The NLP modules and the probabilistic consistency system all use entity and relation names defined in the ontology, to facilitate the integration between these sub-components of the overall FAUST system. When the relations extracted by the IE engine are added to the MLN in PCE, they are prefixed with the term "Mention" to distinguish them from the other predicates in the MLN (see Figure 2). This naming convention allows us to to distinguish the relation extracted by the IE engine with the predicate form we use in PCE to represent that relation in the MLN.

3 Using PCE in FAUST

PCE is used in the FAUST system in a variety of important roles – for anaphora and coreference resolution, for correcting the output of individual IE engines, and also to resolve possible inconsistencies when considering the outputs from multiple IE engines. This section provides insights into some of the mechanisms used in doing MLN-based modeling using PCE in FAUST. The purpose is to illustrate how PCE works together with the StanfordNLP module on real examples to perform coherent and consistent probabilistic inference for answering queries in the NFL domain, using probabilistic first-order rules based on domain knowledge and weighted logical assertions (facts) extracted by StanfordNLP.

3.1 Improving Information Extraction

Let us consider a FAUST system configuration where we have one upstream IE engine, e.g., the StanfordNLP engine. Given a particular query, the IE system extracts entities and

```
### Definitions
sort NFLTeam; sort NFLGame;
sort FinalScore = INTEGER; sort Date;
predicate teamInGame(NFLTeam, NFLGame) indirect;
predicate scored(NFLTeam, FinalScore, NFLGame) indirect;
predicate finalScored(NFLTeam, FinalScore, NFLGame) indirect;
predicate gameWinner(NFLTeam, NFLGame) indirect;
predicate gameLoser(NFLTeam, NFLGame) indirect;
predicate MentionteamScoringAll(NFLTeam, FinalScore) indirect;
predicate MentionteamInGame(NFLTeam, NFLGame) indirect;
predicate MentionhomeTeamInGame(NFLTeam, NFLGame) indirect;
predicate MentionawayTeamInGame(NFLTeam, NFLGame) indirect;
predicate MentiongameDate(NFLGame, Date) indirect;

### Rules
# rules for teamInGame
add [T, S, g] MentionteamScoringAll(T, S) => teamInGame(T, g) 2.0;
add [T, g] MentionteamInGame(T, g) => teamInGame(T, g) 2.0;
add [T, g] MentionhomeTeamInGame(T, g) => teamInGame(T, g) 2.0;
add [T, g] MentionawayTeamInGame(T, g) => teamInGame(T, g) 2.0;

# rule for scored
add [T, S, g] MentionteamScoringAll(T, S) => scored(T, S, g) 3.0;

# rules for finalScored
add [T, S, g] scored(T, S, g) => finalScored(T, S, g) 3.0;
add [T, S1, S2, g] scored(T, S1, g) and scored(T, S2, g) and (S1 > S2)
  => finalScored(T, S1, g) 3.0;
add [T, S1, S2, g] scored(T, S1, g) and scored(T, S2, g) and (S1 > S2)
  => ~finalScored(T, S2, g) 3.0;

# rules for won
add [T1, T2, S1, S2, g] finalScored(T1, S1, g) and finalScored(T2, S2, g)
  and (S1 > S2) and (T1 ~= T2) and teamInGame(T1, g) and teamInGame(T2, g)
  => won(T1, g) 4.0;
add [T1, T2, S1, S2, g] finalScored(T1, S1, g) and finalScored(T2, S2, g)
  and (S1 > S2) and (T1 ~= T2) and teamInGame(T1, g) and teamInGame(T2, g)
  => ~won(T2, g) 4.0;
add [T, g] ~teamInGame(T, g) => ~won(T, g) 3.0;

# rules for won predicate in MLN and MentiongameWinner extracted by Stanford
add [T, g] MentiongameWinner(T, g) => gameWinner(T, g) 1.0;
add [T, g] won(T, g) => gameWinner(T, g) 4.0;
add [T, g] ~won(T, g) => ~gameWinner(T, g) 3.0;

# rules for gameLoser
add [T1, T2, S1, S2, g] finalScored(T1, S1, g) and finalScored(T2, S2, g) and
  (S1 > S2) and (T1 ~= T2) => gameLoser(T2, g) 3.0;
add [T1, T2, S1, S2, g] finalScored(T1, S1, g) and finalScored(T2, S2, g) and
  (S1 > S2) and (T1 ~= T2) => ~gameLoser(T1, g) 3.0;
```

Fig. 2: The jointWinnerLoser MLN used in FAUST.

```
# rules for interactions between gameWinner and gameLoser
add [T, g] gameWinner(T, g) => ~gameLoser(T, g) 3.0;
add [T, g] gameLoser(T, g) => ~gameWinner(T, g) 2.0;
add [T, g] ~gameWinner(T, g) and teamInGame(T, g) => gameLoser(T, g) 1.0;
add [T, g] ~gameLoser(T, g) and teamInGame(T, g) => gameWinner(T, g) 1.0;

### Facts extracted from StanfordNLP
const GreenBayPackers: NFLTeam; const PhiladelphiaEagles: NFLTeam;
const DallasCowboys: NFLTeam; const TampaBayBuccaneers: NFLTeam;
const 3: FinalScore; const 13: FinalScore;
const 34: FinalScore; const 39: FinalScore;
const regular: NFLGame; const y1995: Date;
add MentionteamScoringAll(GreenBayPackers, 39) 1.394;  # p = 0.80
add MentionteamScoringAll(PhiladelphiaEagles, 13) 1.828;  # p = 0.86
add MentionteamScoringAll(GreenBayPackers, 34) 3.535;  # p = 0.97
add MentionteamScoringAll(GreenBayPackers, 3) 4.560;  # p = 0.99
add MentiongameWinner(DallasCowboys) 0.663;  # p = 0.68
add MentionteamInGame(GreenBayPackers, regular) 0.014;  # p = 0.51
add MentiongameDate(regular, y1995) 0.680;

### Query
mcsat_params 100000, 0.01, 5.0, 0.01, 100, 10;
ask[T, g] gameWinner(T, g); ask[T, g] gameLoser(T, g);
```

Fig. 2: The jointWinnerLoser MLN used in FAUST (contd.).

relations and passes them to PCE. Here, we illustrate how PCE interacts with the IE engine through two queries specified in the NFL ontology, namely, the gameWinner and gameLoser.

To illustrate how PCE works with the IE system, we consider a version of the output from StanfordNLP for the document snippet mentioned in Section 1. The following relations are extracted between grounded entities with associated probabilities:

```
teamScoringAll(GreenBayPackers,39) 0.80
teamScoringAll(PhiladelphiaEagles,13) 0.86
teamScoringAll(GreenBayPackers,34) 0.97
teamScoringAll(TampaBayBuccaneers, 3) 0.99
gameWinner(DallasCowboys,game) 0.68
teamInGame(GreenBayPackers,game) 0.51
gameDate(game,y1995) 0.68
```

The IE engine correctly identifies that GreenBayPackers scores 39 (and also 34) and is a team in the game under consideration. However, the IE engine in this case does not infer the correct gameWinner – it suggests that Dallas Cowboys is a winner in the game. These relations are passed to PCE, which uses domain knowledge encoded in domain-specific consistency checks of the form "the winner in a game has to be a player in the game". So even though the IE system suggests that Dallas Cowboys is the winner, PCE uses the other relations extracted by the IE system and its domain-specific rules to override this IE output and infer with high probability that GreenBayPackers is the winner. Here is the output from PCE for the gameWinner query:

```
[T<-GreenBayPackers,g<-game] 0.49: (gameWinner(GreenBayPackers,game))
[T<-DallasCowboys,g<-game] 0.45: (gameWinner(DallasCowboys,game))
[T<-PhiladelphiaEagles,g<-game] 0.40: (gameWinner(PhiladelphiaEagles,game))
[T<-TampaBayBuccaneers,g<-game] 0.39: (gameWinner(TampaBayBuccaneers,game))
```

Here are some important rules in the jointWinnerLoser MLN, which encode some relevant domain knowledge for football, and sports in general.

Rules for "finalScored" relation: If there is one score mentioned in the document, it is the final score with some probability. If it is mentioned that the team had two scores, then the larger score has a higher probability of being the final score.

```
add [T,S,g] scored(T,S,g) => finalScored(T,S,g) 3.0;
add [T,S1,S2,g] scored(T,S1,g) and scored(T,S2,g) and (S1>S2)
  => finalScored(T,S1,g) 3.0;
```

Rules for "won" predicate: If there are multiple final scores of two teams that play in the game, and one final score is higher than the other scores for a team, then the team scoring the higher final score is the winner with a high probability. Also, if a team is not mentioned as a player in a game, it has a high probability of not being the winner.

```
add [T1,T2,S1,S2,g] finalScored(T1,S1,g) and finalScored(T2,S2,g) and (S1>S2)
  and (T1~=T2) and teamInGame(T1,g) and teamInGame(T2,g) => won(T1,g) 4.0;
add [T1,T2,S1,S2,g] finalScored(T1,S1,g) and finalScored(T2,S2,g) and (S1>S2)
  and (T1~=T2) and teamInGame(T1,g) and teamInGame(T2,g) => ~won(T2,g) 4.0;
add [T,g] ~teamInGame(T,g) => ~won(T,g) 3.0;
```

Rules for "gameWinner" relation: Suppose that we infer "won" from the finalScored logic, and StanfordNLP has also extracted a "MentiongameWinner" predicate. In this case, if "won" and "MentiongameWinner" agree, we can declare with high confidence that the team is the winner. If they disagree, then we may want to place higher trust on the winner inferred from the finalScored logic in the MLN. One way to encode this logic could have been using an OR rule R_3 of the following form in the MLN:

```
add [T,g] MentiongameWinner(T,g) or won(T,g) => gameWinner(T,g) 1.0;  # R_1
```

However, in this case we cannot give higher weight to the "won" predicate compared to the "MentiongameWinner" predicate. In order to have this feature of being able to give different weights to the two sources of knowledge about the winner, we could have the following two rules in the MLN instead of R_1:

```
add [T,g] MentiongameWinner(T,g) => gameWinner(T,g) 1.0;  # R_2
add [T,g] won(T,g) => gameWinner(T,g) 4.0;  # R_3
```

If "MentiongameWinner" and "won" disagree, then we give a higher weight 4.0 to the rule R_2 that determines the winner using the finalScored logic in the MLN, and a lower weight 1.0 to the rule R_3 that sets the winner from the "MentiongameWinner" extracted by the IE system.

Having the rules R_2 and R_3 is more flexible than having only R_1. Let us consider an example to convey the intuition for that. If we had one OR rule in the MLN of the form $a \vee b \Rightarrow c$, then we would not be able to put separate weights on inferring c from the predicates a and b. If however we consider a logically equivalent pair of rules $a \Rightarrow c$ and $b \Rightarrow c$, then we can put different weights on the two rules. Having these 2 rules effectively gives us the OR rule, and also allows us to put probabilities on the different literals of the OR. Note that we also have the following rule R_4 in the MLN to reduce the probability of teams from being the final gameWinner if they have not "won" the NFLGame based on the finalScored logic:

```
add [T,g] ~won(T,g) => ~gameWinner(T,g) 3.0; # R_4
```

Rules for "gameLoser" relation: Finally, we added the logic of determining the gameLoser to the rules for determining the gameWinner. We added consistency checks between the predicates "gameWinner" and "gameLoser", as shown in Figure 2. These rules encode checks of the form: (1) the winner cannot be also a loser, and (2) a team that has not won but has played in the game is most probably a loser. Due to the presence of the logic that a team has to be a player in the game to *both* win and lose a game, we cannot simply invert the gameWinner MLN to get the gameLoser MLN. Adding the gameLoser rules to the jointWinnerLoser MLN, as shown in Figure 2, gives us the correct behavior – Philadelphia Eagles is predicted to be the loser with the highest probability in response to the gameLoser query, as shown below:

```
[T<-PhiladelphiaEagles,g<-game] 0.45: (gameLoser(PhiladelphiaEagles,game))
[T<-GreenBayPackers,g<-game] 0.44: (gameLoser(GreenBayPackers,game))
[T<-TampaBayBuccaneers,g<-game] 0.43: (gameLoser(TampaBayBuccaneers,game))
[T<-DallasCowboys,g<-game] 0.39: (gameLoser(DallasCowboys,game))
```

Currently, only the best answer from the marginal probability list is chosen for answering a query. So, we focused more on getting the ordering of the list correct, rather than increasing the difference in actual probabilities. However, if required, the MLN weights can be further fine-tuned to increase this difference as well.

3.2 Consistency between Multiple IE Systems

Another task in FAUST where PCE plays an important role is combining relations extracted by multiple IE systems. In the current FAUST system, we have input coming from multiple IE engines – built by Stanford University, University of Illinois at Urbana-Champaign, University of Massachussets at Amherst, University of Wisconsin at Madison, etc. Currently, the output from the StanfordNLP system has been integrated into FAUST for the NFL domain, and work is ongoing for integrating the other IE systems into FAUST too.

To illustrate how PCE resolves inconsistencies between multiple IE module outputs, let us consider the following text document snippet: *Miami were leading 13-7 at halftime, but a late offensive surge from San Diego led to a dramatic turn of events, with Miami finally getting outscored by 19-28.*

We simulate the output of 3 different IE systems by considering potentially conflicting information extracted from the text snippet by the 3 IE systems:

```
IE 1: teamScoringAll(SanDiego, 28) 0.66
IE 2: teamScoringAll(SanDiego, 28) 0.21, teamScoringAll(Miami, 19) 0.93
IE 3: teamScoringAll(Miami, 13) 0.12, teamScoringAll(SanDiego, 7) 0.27
```

Since FAUST uses multiple IE systems, it is imperative to resolve such potential conflicts between the outputs of the IE systems. When all of these IE relations are added as probabilistic facts to the jointWinnerLoser MLN outlined in Section 3.1 and MC-SAT inference is performed, PCE is able to correctly infer that San Diego is the gameWinner and Miami is the gameLoser.

Learning weights: For the jointWinnerLoser MLN, we tuned the weights on the formulas using a development set. We are currently working on developing and implementing more efficient weight learning algorithms [8, 7, 17].

Table 1: Comparison of StanfordNLP and PCE.

Module	Precision	Recall	F-measure
StanfordNLP	0.438	0.202	0.277
PCE	0.441	0.384	**0.411**

4 Experimental Results

We ran experiments to compare the output of the StanfordNLP system with the output from the joint StanfordNLP+PCE system. The goal of the paper is to demonstrate empirically how IE and probabilistic reasoning work together jointly in the FAUST architecture to produce more desirable results in the example NFL domain, compared to just using the StanfordNLP engine alone. For ease of notation, we use the terminology *StanfordNLP* to mean *output from only the StanfordNLP system*, and the terminology *PCE* to mean *output from the joint StanfordNLP+PCE system*. Note that here we show the comparison of the PCE output with the standalone StanfordNLP system output only with respect to the gameWinner relation. Also, the StanfordNLP is essentially a baseline IE system augmented with hard rules – so in essence, the focus of our experiments is to show the improvement that global inference techniques (e.g., PCE) can give when used in conjunction with local information extraction methods augmented with deterministic rules (e.g., StanfordNLP).

We ran the comparison experiment on a corpus consisting of 500 NFL news documents. These documents had descriptions of games, with the game winners either explicitly extracted as MentiongameWinner relations by the StanfordNLP system or implicitly implied through extracted final scores, which were then used in the jointWinnerLoser MLN for inferring the correct gameWinner and gameLoser. We used a separate development set for tuning the rules (structure) and weights of the jointWinnerLoser MLN, which was independent of the corpus used to evaluate the performance of the MLN.

The probabilistic relations extracted by StanfordNLP system were then used as probabilistic facts by the jointWinnerLoser MLN, as outlined in Section 2. Table 1 shows that the recall of PCE is significantly higher than StanfordNLP, leading to substantially higher F-measure. Note that in this case, precision, recall and F-measure are defined as:

$$\text{Precision} = \frac{\text{Number of correct gameWinner predictions}}{\text{Total number of gameWinner predictions}}$$

$$\text{Recall} = \frac{\text{Number of correct gameWinner predictions}}{\text{Actual number of gameWinner results in the corpus}}$$

$$\text{F-measure} = \frac{2}{\text{1/Precision + 1/Recall}}$$

When StanfordNLP extracts a game winner from the text incorrectly, it is mostly because in this case StanfordNLP does the extraction at the sentence level, without doing global inference. Let us consider some examples where PCE improves over StanfordNLP output, to demonstrate why PCE gets a better F-measure than StanfordNLP:

1. Consider the sentence: *After boasting all week that the Giants couldn't move on their league-leading unit, Ray Lewis and the Ravens did what they said they would, beating New York 34-7 for their first Super Bowl victory.* StanfordNLP extracts Giants as the

game winner from the above sentence. PCE on the other hand uses the information extracted by StanfordNLP, that Baltimore Ravens scored 34 and NewYork Giants scored 7, to correctly infer using the jointWinnerLoser MLN that Baltimore Ravens is the winner.

2. Consider the sentence: *Gannon threw for 286 yards and three touchdowns and ran for another score Sunday to lead the Raiders to a 41-24 victory over the Tennessee Titans in the National Football League's AFC conference championship game.* StanfordNLP extracts that Raiders scored 41 and Titans scored 24, but the hard rules in the StanfordNLP system are not able to infer a game winner. PCE uses the extracted scores and the probabilistic rules in the jointWinnerLoser MLN to identify that the higher scoring team Oakland Raiders were the game winner.
3. One document in the corpus talks about a game where PhiladelphiaEagles beat WashingtonRedskins 34 to 21. In the context of this game, it also talks about some previous games. StanfordNLP identifies multiple teams, including the correct game winner PhiladelphiaEagles, as possible game winners for the game in question. However, PCE correctly infers that PhiladelphiaEagles is the only winner in the game under consideration, by doing collective inference across the whole document.

5 Related Work

In this paper, we build an MLN for solving tasks (find game winner and loser) based on information extracted from natural language text. The problem of doing information extraction simultaneously with some other inference task (e.g., segmentation, entity resolution) has been studied previously in the information extraction literature – we discuss a representative sample of that literature in this section.

Roth et al. [18] proposed representing and learning propositional relations for information extraction. Poon et al. [16] proposes a technique for jointly doing information extraction along with segmentation and entity resolution, using MLNs. Bunescu et al. [2] uses a relational Markov network model for doing collective information extraction across multiple sentences. Finkel et al. [5] introduces long-range interactions in information extraction models like conditional random fields (to get skip-CRFs), and propose efficient inference using Gibbs sampling. Mann et al. [9] can do information extraction across multiple documents using a technique of cross-field bootstrapping, which fuses the information extracted across different documents. McCallum et al. [10] proposed using relational probabilistic models for doing information extraction along with data mining.

The Holmes system [19] uses rules as Markov logic Horn clauses to carry out textual inference over documents. Textual entailment [15, 4] is related to our work though our queries are not posed in natural language. Bryl et al. [1] exploit background semantic knowledge in resolving co-references in text. Yao et al. [22] use probabilistic inference on undirected graphical models constructed for joint relation and entity extraction from text. Niu et al. [13] have worked on scaling MLN inference using database techniques, while Shavlik et al. [20] scale MLNs by pre-processing to reduce the size of the grounded network.

Many statistical relational models, e.g., Relational Dependency Networks [12], Probabilistic ILP [3], Probabilistic Relational Models [6], First Order Conditional Influence Language [11], have been applied to information extraction and other NLP tasks. The novelty of our approach lies in using PCE as a harness in FAUST to join multiple NLP systems, such that we can use probabilistic first-order semantic rules to ensure domain-specific global consistency between facts extracted by one or more NLP systems.

6 Conclusions and Future Work

We describes an MLN-based probabilistic relational model (PCE) in the FAUST system for machine reading, which takes information from multiple IE engines and does overall consistency checks based on domain knowledge. Our experimental evaluation showed how PCE improves performance compared to individual IE engines, e.g., StanfordNLP.

The corpus of 500 NFL documents are an initial representative sample from the Machine Reading project – we are in the process of applying these techniques to much larger datasets, e.g., MR-KBP (Knowledge Base Population) and IC (Intelligence Community), where there are more than a million documents. In future work, we would like to learn the weights and structure of the MLNs automatically from data available in the domain. During this work, output from only one IE engine (StanfordNLP) was available to us for the NFL domain in our system. So, we had to simulate a proof-of-concept of how PCE can resolve inconsistencies across multiple IE outputs, based on actual IE system outputs that we had seen/analyzed so far. In future, when multiple IE system outputs become available, we would like to have inconsistency resolution across multiple input IE systems in our experimental evaluation.

Acknowledgments: The authors gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. The authors would also like to thank the Stanford CoreNLP team (especially Christopher Manning, Mihai Surdeanu, David McClosky, and Dan Jurafsky), the SRI team (especially Lynn Voss, Brent Ellwein, David Israel, Hung Bui), the Wisconsin team (especially Jude Shavlik), and Sharad Shankar.

References

1. Bryl, V., Giuliano, C., Serafini, L., Tymoshenko, K.: Using background knowledge to support coreference resolution. In: ECAI (2010)
2. Bunescu, R., Mooney, R.J.: Collective information extraction with relational Markov networks. In: ACL (2004)
3. De Raedt, L., Kersting, K.: Probabilistic Inductive Logic Programming, chap. Probabilistic Inductive Logic Programming. Springer (2008)
4. De Salvo Braz, R., Girju, R., Punyakanok, V., Roth, D., Sammons, M.: An inference model for semantic entailment in natural language. In: IJCAI (2005)
5. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: ACL (2005)
6. Getoor, L., Friedman, N., Koller, D., Pfeffer, A.: Learning Probabilistic Relational Models, chap. Relational Data Mining. Springer-Verlag (2001)
7. Huynh, T.N., Mooney, R.J.: Online max-margin weight learning for Markov logic networks. In: SDM (2011)
8. Lowd, D., Domingos, P.: Efficient weight learning for Markov logic networks. LNCS pp. 200–211 (2007)
9. Mann, G.S., Yarowsky, D.: Multi-field information extraction and cross-document fusion. In: ACL (2005)
10. McCallum, A., Jensen, D.: A note on the unification of information extraction and data mining using conditional-probability, relational models. In: IJCAI Workshop on Learning Statistical Models from Relational Data (2003)
11. Natarajan, S., Wong, W.k., Tadepalli, P.: Structure refinement in first order conditional influence language. In: Proc. Workshop on Open Problems in SRL (2006)
12. Neville, J., Jensen, D.: Relational dependency networks. JMLR pp. 653–692 (March 2007)
13. Niu, F., Re, C., Doan, A., Shavlik, J.: Tuffy: Scaling up statistical inference in Markov logic networks using an rdbms. In: VLDB (2011)
14. Owre, S., Shankar, N.: PCE User Guide, Technical manual, version 1.0. CSL, SRI International (July 2009)
15. de Paiva, V., Bobrow, D.G., Condoravdi, C., Crouch, D., King, T.H., Karttunen, L., Nairn, R., Zaenen, A.: Textual inference logic: Take two. In: Workshop on Contexts and Ontologies: Representation and Reasoning, ICMUC (2007)
16. Poon, H., Domingos, P.: Joint inference in information extraction. In: AAAI (2007)
17. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
18. Roth, D., Yih, W.t.: Relational learning via propositional algorithms: An information extraction case study. Tech. Rep. UIUCDCS-R-2001-2206, UIUC CS Department (2001)
19. Schoenmackers, S., Etzioni, O., Weld, D.S.: Scaling textual inference to the web. In: EMNLP (2008)
20. Shavlik, J., Natarajan, S.: Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In: IJCAI (2009)
21. Surdeanu, M., McClosky, D., Manning, C.: Customizing an information extraction system to a new domain. In: RELMS (2011)
22. Yao, L., Riedel, S., McCallum, A.: Collective cross-document relation extraction without labelled data. In: EMNLP (2010)