Understanding Robocup-Soccer Narratives

Hannaneh Hajishirzi and Eyal Amir
University of Illinois at Urbana-Champaign
{hajishir, eyal}@illinois.edu

Abstract

We present an approach to map Robocup-soccer narratives (in natural language) to a sequence of meaningful events. Our approach takes advantage of an action-centered framework, an inference subroutine, and an iterative learning algorithm. Our framework represents the narrative as a sequence of sentences and each sentence as a probability distribution over deterministic events. Our learning algorithm maps sentences to meaningful events without any annotated labeled data. Instead, it uses a prior knowledge about event descriptions and an inference subroutine to estimate initial training labels. The algorithm further improves the training labels at next iterations. In our experiments we demonstrate that with no labeled data our algorithm achieves higher accuracy compared to the state of the art that uses labeled data.

Introduction

Mapping sentences of a narrative to a sequence of meaningful events is an important, but very difficult, problem in Natural Language Processing (NLP) and linguistics. Finding such mapping automatically alleviates the need of human participations in many applications. The examples of those applications include, but not limited to, commonsense query answering, help desks, and navigational systems. For example, automatic reading comprehension systems can answer semantic questions about the narrative using this meaningful representation. The examples of such questions are “Where is the ball at every time step of the game?”. Current narrative understanding systems cannot answer such questions because they look for a response that lies in the text.

Recently, there have been some approaches that map natural language sentences to meaning representations. Most of these approaches (e.g., (Zettlemoyer & Collins 2009; Chen, Kim, & Mooney 2010; Kate & Mooney 2007)) use some sort of annotated data to train a classifier that identifies meaning representations. Some other approaches (e.g.,(Branavan *et al.* 2009; Vogel & Jurafsky 2010; C. Matuszek & Koscher. 2010)) provide indirect supervision for the classifier by interacting with a physical dynamic environment to validate the selected meaning representations. It is often very hard (sometimes infeasible) to either build human annotated data or have access to an interactive environment.

In this paper, we introduce an iterative learning algorithm that maps Robocup-soccer narratives to meaningful events with no annotated labeled data. Instead, we use a prior knowledge (in terms of few event specifications) that is much easier to construct compared to annotating every sentence of the narrative. Our iterative learning approach achieves higher accuracy than (Chen, Kim, & Mooney 2010) that uses labeled data. This improvement is possible due to the representation, the inference subroutine that applies event specifications to estimate initial labels, and iterative learning that improves the quality of estimated labels.

We use a powerful representation to model Robocup-soccer narratives. This representation (extension of probabilistic situation calculus (Reiter 2001)) allows explicit tracking of the change in the state of the narrative. In this representation, sentences of the narrative are mapped to probabilistic transitions which give rise to deterministic events according to a probability distribution. Current state of the narrative affects the probability that each event has occurred. Deterministic events in turn give rise to world changes that also depend on the current state.

We use an inference subroutine to keep track of the current state of the narrative by applying event specifications. We use this subroutine together with NLP tools to estimate initial labels for our iterative learning approach. The labels are initially estimated by finding feasible events (consistent with the state of the narrative) and has low edit distance with the verb of the sentence.

Our iterative learning approach computes the score of choosing an event corresponding to a sentence given the current state of the narrative. Our algorithm builds training examples as triplets of sentence, event, and the current state from event histories. Event histories are constructed from sampling events for sentences and applying inference to update the current state. At each iteration, our approach trains a classifier based on the current training labels which will be further improved using the current classifier.

We use our framework together with learning and inference algorithms to analyze Robocup soccer games. We demonstrate that using prior knowledge about events improves the accuracy of mapping narratives meaningful events compared to (Chen, Kim, & Mooney 2010). What is more, we show that this event knowledge alleviates the need of manually labeled data as we show improvement over (Chen, Kim, & Mooney 2010) where they use labeled data.

Representation

In this section we describe Probabilistic Action Model (PAM) representation to model relational dynamic domains. We later use PAM to represent robo-soccer narratives.

PAM is a framework for representing dynamic systems and consists of two main parts: (1) a prior knowledge including a probability distribution over state variables at the initial time step and commonsense knowledge about specifications of events (2) a transition model to represent the dynamics of the system. In PAM the transition model is represented naturally as a probability distribution over different deterministic events. PAM is also augmented with elements of first-order logic to allow compact representation. This representation instantly resembles a narrative where a sentence describes an uncertainty among different meanings and every meaning identifies a deterministic event that occurs in the domain. More specifically PAM representation is defined as follows.

Definition 1. A PAM $(\mathcal{L}, \mathit{EA}, \mathit{PA}, P_0)$ consists of a

- language $\mathcal{L}$ describing the elements of the representation (Def. 2)
- set of effect axioms *EA* for deterministic events (Def. 3)
- transition modeled with probabilistic actions *PA* (Def. 4)
- prior probability P_0

Language: The language $\mathcal{L}$ of PAM is almost similar to the language of predicate logic. The language consists of a set of constants, variables, predicates (called fluents), and deterministic events. In this language every constant and every variable has a specific *type*, and every predicate and deterministic event has a *schema*. Specifically, $f(\tau_1, \ldots, \tau_k)$ (or $f(\vec{\tau})$) is a predicate schema where f is a predicate symbol and $\tau_1, \ldots, \tau_k$ are types.

Definition 2. The language $\mathcal{L}$ of a PAM is a tuple $\mathcal{L} = (\mathit{Ty}, \mathit{I}, \mathit{Itype}, \mathit{V}, \mathit{Vtype}, \mathit{F}, \mathit{DA})$ consisting of

- a set of types *Ty*
- a set of constants *I* and a function *Itype* which maps every constant to a type $\mathit{Itype} : I \to \mathit{Ty}$
- a set of variables *V* and a function $\mathit{Vtype} : V \to \mathit{Ty}$
- a set of predicate variable schemas *F*
- a set of deterministic event schemas *DA*

The predicates in the language are called fluents as their truth values change over time. In PAM grounding of a fluent $f(\vec{x})$ is defined as replacing each variable in $\vec{x}$ with a constant. Accordingly, a world state is defined as a full assignment of $\{\mathit{true}, \mathit{false}\}$ to all the groundings of all the fluents in F. However, it is generally the case that, at any particular time step, the values of many fluents are not known. Therefore, we introduce the notion of partial states. The state of the narrative is a conjunction of fluents that are *true* at a specific step.

Deterministic Events: In PAM deterministic event specifications are in the form of effect axioms.

Definition 3. Let $\mathit{da}(\vec{x})$ be a deterministic event schema and $\mathit{Effect}(\vec{x})$ and $\mathit{Precond}(\vec{x})$ be conjunctions of fluent schemas whose variables appear in $\vec{x}$. The following represents the effect axioms for *da*.

- $\mathit{da}(\vec{x})$ **initiates** $\mathit{Effect}(\vec{x})$.
- $\mathit{da}(\vec{x})$ **initiates** $\mathit{Effect}(\vec{x})$ **when** $\mathit{Precond}(\vec{x})$.

The semantics of the above definition is as follows: For every time step t if *Precond* holds at time t, then the deterministic event *da* initiates *Effect* at time $t + 1$.

Probabilistic Transitions: The transition model in PAM is represented as a probability distribution of choosing deterministic events conditioned on the current state.

Definition 4. Let $\psi_1(\vec{x}), \ldots, \psi_N(\vec{x})$ be partial states partitioning the world. The following is the *transition model pa*: $pa(\vec{x})$ **produces**

- $da_1^1, \ldots, da_1^M$ **with** $p_1^1, \ldots, p_1^m$ **when** $\psi_1(\vec{x})$
- $\ldots$
- $da_N^1, \ldots, da_N^M$ **with** $p_N^1, \ldots, p_N^m$ **when** $\psi_N(\vec{x})$

where $\vec{x}$ are arguments of *pa*, and p_J^i is the probability of choosing event da_J^i in the partition ψ_J with probability p_J^i.

We use a Robocup-soccer dataset and show how to represent English narratives of soccer games using PAM representation. We consider the robocup-soccer narrative as a sequence of sentences extracted from the narrative. In fact, the narrative model $\mathit{Tx} = (\mathit{T}, \mathit{OA})$ for *PAM* is a sequence with length *T* of sentences $\mathit{OA} = \langle a_1, \ldots, a_T \rangle$ where a_t is the t^{th} sentence. In fact each sentence in the narrative is mapped to the PAM transition model which is a probability distribution over deterministic events.

We build PAM representation corresponding to the narratives by defining the language of the PAM followed by manually constructing effect axioms of deterministic events. We further learn the transition model automatically by using our iterative learning approach (Section).

We outline the PAM language as follows. We define two types *team* and *player*. We instantiate *team* to two instances "Purple" and "Pink" and *player* to 22 instances. Fluents of PAM include different playmodes of the game such as *atOffside*, *kickOff* in the dataset. We add a new predicate called *holding(player)* to show that *player* holds the ball at a specific time step.

We introduce prior knowledge by manually building effect axioms for each deterministic event. We augment deterministic events *DA* of PAM with a deterministic "noise" event called *nothing* that has no preconditions and no effects. The reason of adding this noise action is that (1) the given narrative is not always consecutive and there are some events missing (2) no meaning representations for a specific sentence is available in the prior knowledge.

Basic Inference by Consistency Checking

In this section we outline the basic *inference* algorithm that is a Viterbi-like (Rabiner 1989) dynamic programming approach that finds the most likely sequence of events corresponding to an English narrative. Here we assume that the transition model of PAM is available. Next (Section) we introduce an iterative learning subroutine that automatically computes the transition probability corresponding to every sentence in the narrative. This probability depends on the current state of the narrative while learning this probability is achieved with no labeled data.

The algorithm *Inference* takes as input the narrative $OA = \langle a_1, \ldots, a_T \rangle$ and the PAM representation with deterministic events *DA* and the transition probability *Pa*. The algorithm returns the Viterbi sequence $\langle ev_1, \ldots, ev_T \rangle$ as an approximation of the most likely event sequence corresponding to the narrative given by the following recurrence relations.

$$
\begin{aligned}
&V_{1,da} = \mathit{Pa}(da|a_1, s_0),\ S_{1,da} = \mathit{Prog}(s_0, da),\ \mathit{Path}_{1,da} = [da] \\
&V_{t,da} = \mathit{Pa}(da|a_t, s_{t-1}) + V_{t-1,ev} + l_{s_{t-1},da} \\
&S_{t,da} = \mathit{Prog}(s_{t-1}, da)), \mathit{Path}_{t,da} = \mathit{Path}_{t-1,ev} + [da] \qquad (1)
\end{aligned}
$$

where $ev = \arg\max_{da \in DA}(V_{t-1,da})$, $s_{t-1} = S_{t-1,ev}$, and $l_{s,da}$ is a loss function.

Here $V_{t,da}$[1] shows the value of the most likely event sequence $Path_{t,da}$ for the first t sentences, $S_{t-1,da}$ shows the current state of the narrative at time $t-1$, . The *Inference* algorithm initializes the value of the path with the probabilities of events for the first sentence. Then, at each time step the algorithm integrates the probability of event da and the maximum value derived for the step $t-1$. If the preconditions of da is not consistent with the current state s_{t-1} we penalize the value of choosing this event using a loss function $l_{s_{t-1},da}$ which is a real number between 0 and 1. Current state $S_{t,da}$ is derived by *Progressing* state s_{t-1} with event da. The Viterbi path $Path_{t,ev}$ updated by keeping a pointer to the previous selected event in the recursive step. Finally, the most likely event sequence is derived as: $ev_T = \arg\max_{da \in DA}(V_{T,da}), ev_{1..T-1} = Path_{da_t,t}$.

Progress subroutine $Prog(ev, s_{t-1})$ takes as input an event da and the current state of the system s_{t-1} and returns the updated state s_t if the preconditions of the event da is consistent with s_{t-1}. It then updates the current state of the system by applying the effect axioms of the event da.

This viterbi-like approach does not return the exact most likely event sequence as it depends on a loss function at every step that the event is inconsistent. To return the exact sequence, we need to consider all the possible event sequences of the forest corresponding to the narrative. However, it is not feasible as the narratives are either too long or have high branching factor. Only for the 2001 game for the first 100 comments it took about 50 Gig memory. The main of goal of this paper is not to improve the accuracy of the inference algorithm. We want to show that an efficient inference algorithm to compute the current state of the narrative, careful design of representation, and an iterative learning algorithm achieves us significantly better results compared to the state-of-the-art.

There have been several works in the literature on exact query answering about a narrative as a sequence $\langle a_1, \ldots, a_T \rangle$ (e.g., (Baral & Tuan 2002)). The main idea is to build a forest to maintain all, and only, feasible interpretations of the narrative in an online fashion. However, this approach is not feasible long narratives with large branching factor.Sampling possible deterministic events of the narrative (Hajishirzi & Amir 2008) is faster than the exact computation, but it is still too expensive for our problem. The algorithm approximates the inference by generating samples among possible deterministic executions of the given narrative.

Learning Transition Model

In this section we introduce our iterative learning approach *learn-inference* that computes the transition probability corresponding to every sentence in the narrative without any annotated labeled data. Instead, our algorithm uses a prior knowledge about event effect axioms with an inference subroutine to update the current state of the narrative. The algorithm first trains a classifier using an iterative EM like learning algorithm (similar to EM-like Multiple Instance Learning (Babenko, Yang, & Belongie 2009)). This algorithm

[1]Notice that V is not a probability function.

```
Algorithm 1. TrainIter($Tx, PAM, K$)
• Input: training narrative Tx, effect axioms EA, language PAM.L
1. Repeat until convergence
   (a) $(ev_i, a_i, s_i)_{i:1..N} \leftarrow HistoryGenerator(Tx, EA, K)$
   (b) for $i$ : 1 to $N$
       i. $\vec{F}_i \leftarrow FeatureGenerator(ev_i, a_i, s_i, PAM.L)$
       ii. $l_i \leftarrow VoteGenerator(ev_i, a_i, s_i, EA)$
   (c) $\vec{W} \leftarrow Classifier(\vec{F}_{i:1..N}, l_{i:1:N})$
Algorithm 2. learn-Inference(Train $Tx_{1..3}$, Test $Tx$, $PAM$)
• Input: Train narratives $Tx_{1..3}$, Test narrative $Tx$, $PAM$
1. $\vec{W} \leftarrow Train(Tx_{1..3}, PAM)$
2. $PAM.Pa \leftarrow Test(Tx[1], true)$
3. $Inference(Tx_4, PAM)$[a]
```

[a]*PAM.Pa* is updated at every step t of the *inference* subroutine using $Test(Tx[t], s_{t-1})$.

Figure 1: *learn-Inference* algorithm to compute the transition model corresponding to each sentence in the test narrative.

trains a classifier based on a set of initial approximate labels derived by applying an inference algorithm given the event specifications. The algorithm further iterates to improve training labels by applying the learned classifier over training data. Finally, the algorithm applies the learned classifier over the narrative to compute the transition probability assigned to every sentence.

Training: *TrainIter* algorithm takes as input a robo-soccer narrative with sentences $\langle a_1 \ldots a_T \rangle$, the language of the corresponding PAM, and the effect axioms for deterministic events. At each iteration, the algorithm first generates training examples of the form $e_i = (da_t, a_t, s_{t-1})$ using *history generator*. It then assigns features to each example using *feature generator*. Then, it uses *vote generator* to assign approximate labels to each example e_i. The votes are initialized using event axioms and inference subroutine. Finally, the *classifier* generates a weight vector according to the current training examples and their votes. These steps are iterated until convergence. In next iterations the votes are improved using the classifier at that time by using *score generator* to computes the score of the examples. Here we describe each modules in details.

Testing: The input to test module is a sentence s with the current state of the narrative st. In turn it returns the probability of different events corresponding to the sentence. The algorithm first uses the arguments of the sentence and grounds all the possible event types ev_i associated with the sentence. It then normalizes the scores of examples (ev_i,s,st).

Finally, *learn-inference* algorithm applies *inference* subroutine to the probabilities derived from the testing module. Notice that the current states are updated through the *infernce* subroutine after choosing the best event at every time step.

Experiments

We use Robocup dataset from (Chen, Kim, & Mooney 2010). This dataset is based on four championship games of Robocup simulation league from 2001 to 2004. We generate the PAM representation corresponding to this dataset according to Section where we manually build effect axioms for deterministic events. We examine our algorithms on this dataset to evaluate the effect of adding prior knowledge (event specifications) to find the most likely event sequence corresponding to the narrative. We compare our algorithm with different baselines and *wasper-gen-igsl-meteor*(WGIM) which is the most accurate algorithm from (Chen, Kim, & Mooney 2010) which uses annotated labeled data from even logs of the games in the form of a mapping between sentence and the events that appear in the event log of the game within the 5 timesteps that the comment is recorded.

We first compute the performance of algorithms to find the most likely event sequence corresponding to a narrative. For evaluation purposes only, we use manually constructed gold-standard labels from (Chen and Mooney) where each sentence is matched to the correct event. We evaluate the accuracy of resulting event sequence by computing the proportion of the events that have been correctly assigned to the sentences. We conducted our experiments in two settings of with labeled data and without labeled data.

Without labelled data: We apply *learning-inference* to map a soccer narrative to an event sequence and show that the resulting event sequence is more accurate compared to baselines and (Chen and Mooney) where they use annotated data. With respect to our learning algorithm, we train on three Robocup games (e.g., 2001, 2002, and 2003) and test on the last game (e.g., 2004). We then run *Learn-inference* and compute the weight vector. The average number of training examples per iteration is around 700. Notice that the votes are initialized using event specifications and are improved in each iteration.

We compare our iterative learning algorithm (*learn-inference*) with no annotated data by different baselines and *WGIM* (Chen, Kim, & Mooney 2010) that uses labeled data. The first baseline is (*inference*) where the possible events for every sentence are among the ones that have low edit distance with a word in the sentence. The second baseline (Entity) which selects the events corresponding to a sentence if they have matching number of arguments. The third baseline (Entity-inference) combines inference subroutine *inference* and *Entity*.

Table 1 shows the comparison with the gold-standard labels for finding the best event sequence. The results demonstrate that our *learn-inference* algorithm has higher accuracy compared to baseline algorithms with no labeled data. What is more, our results show improvement over WGIM which uses labeled data. These results validate our intuition that prior knowledge (in terms of event specifications) gives us enough knowledge to forgo the expensive manual data annotation in supervised learning approaches.

With labelled data: Here we show that if we augment Chen and Mooney's approach with knowledge abouft events we significantly improve their accuracy. For this experiment, we apply *inference* where transition model is derived from (Chen and Mooney's) approach. Table 1 shows that by applying inference and keeping two consecutive states consistent improves the results of (Chen, Kim, & Mooney 2010) to find the most likely event sequence. Notice that they do not check if the selected event is consistent with the rest of the sequence.

Approach	2001	2002	2003	2004	Avg.
Prior knowledge (no annotated data)					
Learn-Inference	**.739**	**.647**	**.867**	.653	**.728**
Inference	.688	.464	.640	.454	.573
Entity	.625	.564	.718	**.720**	.648
Entity-Inference	.639	.577	.733	.708	.656
Annotated labeled data					
(Chen,Mooney)-inference	**.767**	**.721**	.638	**.798**	**.734**
(Chen, Mooney)	.721	.664	**.683**	746	.703

Table 1: (*top*) Comparing our approach *learn-inference* with other approaches to find the most likely event sequence corresponding to Robocup narratives.

Conclusions and Future Work[2]

In this paper we introduce an approach to map Robocup-soccer narratives to sequences of meaningful events without anny annotated labeled data. In this approach we alleviate the need of labeled data and automatically generate labels for training examples using prior knowledge about events together with the careful design of representation and an inference algorithm. We further use an iterative learning approach and improve the accuracy of labels. We show that this approach improves the state-of-the-art which uses labeled data. Our approach works well when the narrative is given in a temporal form (like reports, instructions, and stories) where the state changes over time. A very important application of our work is inferring or predicting missing events in the narrative. We can apply the inference algorithm and find the missed sequence of events when two consecutive sentences are inconsistent.

References

Babenko, B.; Yang, M.-H.; and Belongie, S. 2009. Visual tracking with online multiple instance learning. In *CVPR*.

Baral, C., and Tuan, L. 2002. Reasoning about actions in a probabilistic setting. In *AAAI*.

Branavan, S.; Chen, H.; Zettlemoyer, L.; and Barzilay, R. 2009. Reinforcement learning for mapping instructions to actions. In *ACL-IJCNLP*, 82–90.

C. Matuszek, D. F., and Koscher., K. 2010. Following directions using statistical machine translation. In *HRI*.

Chen, D.; Kim, J.; and Mooney, R. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *JAIR* 37:397–435.

Hajishirzi, H., and Amir, E. 2008. Sampling first order logical particles. In *UAI*.

Kate, R. J., and Mooney, R. J. 2007. Learning language semantics from ambiguous supervision. In *AAAI*, 895–900.

Rabiner, L. R. 1989. A tutorial on HMM and selected applications in speech recognition. *IEEE* 77(2).

Reiter, R. 2001. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Vogel, A., and Jurafsky, D. 2010. Learning to follow navigational directions. In *ACL*.

Zettlemoyer, L., and Collins, M. 2009. Learning context-dependent mappings from sentences to logical forms. In *ACL-IJCNLP*, 976–984.

[2]This work was supported by NSF CAREER 05-46663 grant.