Using an Episodic Memory Module for Pattern Capture and Recognition

Dan Tecuci and Bruce Porter
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
{tecuci,porter}@cs.utexas.edu

Abstract

We propose an episodic memory-based approach to the problem of pattern capture and recognition. We show how a generic episodic memory module can be enhanced with an incremental retrieval algorithm that can deal with the kind of data available for this application. We evaluate this approach on a goal schema recognition task on a complex and noisy dataset. The memory module was able to achieve the same level of performance as statistical approaches and doing so in a scalable manner.

1 Introduction

A growing class of AI applications rely on capturing and recognizing complex patterns in their data: crime and terrorism prevention, tax fraud detection, trend assessment, etc.

The two main approaches to building these applications are memory-based (classify a new situation by using similarity with prior cases) and generalization-based (derive general classification rules). The purpose of a memory is to store relevant prior experience so that they are available for future use and to do this in a time and space efficient manner. This corresponds to the capture of patterns and their recognition in newly observed data.

However, classical memory-based approaches (like case-based reasoning) cannot be directly applied to this domain. This is due to important differences in the *type of patterns* captured (description of situations/entities in CBR vs. temporal sequences of complex events), the *dynamic aspect of patterns* (events have applicability conditions and consequences) and the *incremental availability of data* (due to limited observation and to the fact that complex events unfold over time).

We propose to use a **episodic memory** for the problem of pattern-based analysis. Such a memory has all the required characteristics: it organizes *temporally ordered events*, these events are *dynamic* (i.e. they change the state of the world) and they are *observed incrementally*. Capture and recognition of past events are the basic processes of an episodic memory.

Prior AI applications exhibit only one of these characteristics: [Kolodner, 1984] designed a memory for organizing events by when and where they happened, but it does not do episodic recognition; the Basic Agent ([Vere and Bickmore, 1990]) employed two episodic memories, one made up of simple recognizers (e.g. repetition) and the other one for guiding the planner's backtracking mechanism; [Ram and Santamaria, 1997] recorded raw data from prior actions to improve navigation; [Nuxoll and Laird, 2004] designed the most advanced cognitive model of an episodic memory, addressing all the functional stages proposed by [Tulving, 1983]. Its main limitations are the flat episode organization which results in retrieval time linear in the number of stored episodes.

We have designed a **generic episodic memory module** that can be applied for a multitude of tasks in various domains. We do not propose complete solutions for problem solving in these domains, rather the episodic memory will have a supporting role in solving such problems. Encapsulating much of the complexity needed to build such a system in its memory subsystem can pay off by simplifying other components.

We augment this memory module with an incremental retrieval algorithm, suitable for the task of pattern recognition, and evaluate its performance on a plan recognition task on a synthetic dataset in the logistics planning domain.

2 Related Work

There has been a lot of related work in plan recognition ([Schmidt *et al.*, 1978]) - the act of reasoning from a set of observed actions in order to infer the goal, a possible next action, or a complete plan (a sequence of steps for achieving a plan).

Both symbolic and probabilistic approaches to plan recognition have been proposed ([Kautz, 1997], [Pollack, 1990]). Criticism focuses mainly on their inability to deal with error recovery and incremental recognition([Woods, 1990]).

In this paper we show that our generic episodic memory module can address some of these problems. An incremental retrieval algorithm that works with our memory module is proposed. Error recovery is dealt with by keeping all plausible hypotheses available and only eliminating those that become inconsistent with the observed data. By storing only relevant patterns we reduce the size of the search space that needs to be examined at retrieval time.

3 A Generic Memory Module for Events

Most of the tasks an intelligent system accomplishes can be represented as events, so, in order to store them, a memory for events is needed. There have been numerous attempts at building some form of memory into an intelligent system (the field of case-based reasoning, Soar) but most suffered from being limited to a particular domain task.

Our proposal is too develop a *generic* memory module that could be used in a variety of applications and for a variety of tasks. Such a memory is intended to be used in connection with an a application module that would supply necessary domain dependent interface functions.

3.1 General Memory Requirements

Remembering the past is necessary but not sufficient: it needs to be done in a timely manner. Intelligent systems need to *efficiently* organize and search their memory. They also need to retrieve items *relevant* to the situation at hand. This requires a *flexible match* between the previous experience and and the new situation. Such a memory needs to be *content-addressable* such that an external stimulus can directly index relevant knowledge in memory. These internal memory requirements have to be combined with the external ones, of which scalability and robustness are the most important.

Such a generic episodic memory module should provide:

- a representation of generic episodes that supports reuse.
- an organization of such episodes that can accommodate a large number of them and is able to retrieve them efficiently.
- generic mechanisms for deciding what to store from an episode and what features should be used in retrieval; these mechanisms should enforce (partial) reuse of past memories.
- scalable and generic retrieval mechanisms.
- generic update mechanisms for when new experience becomes available.

3.2 The Design of the Generic Memory Module

Episode Representation

A generic episodic memory needs to have a representation for a generic episode. We define an episode as a sequence of actions with a common goal. A semantic memory (i.e. a knowledge base) will provide knowledge about domain-specific actions, their applicability conditions and the effects and goals they achieve.

We have identified three major classes of systems that can benefit from using a memory module:

memory-based planning - devise a plan (a sequence of actions) that accomplishes a given goal. Using a memory of past plans (along with their results), an agent can save the work by adaptating prior plans to achieve similar goals and can avoid plan failures either by recognizing them early on or by recalling a plan repair that worked previously.

memory-based diagnosis - diagnose a malfunction based on its symptoms. Memory can help organize diagnoses by their symptoms, contexts in which they manifest, necessary treatments, etc.

memory-based recognition - recognize a sequence of events achieving some goal (usually undesired). Memory organizes plans by their actions, goals they achieve, and failures they encounter.

Based on the characteristics of these three classes of applications, we propose that a generic episode have three parts:

context - the general setting in which some episode happened; for some applications this might be the goal of the episode (e.g. planning), representing the desired state of the world after the episode is executed.

contents - what the episode consists of: the ordered set of events that make up the episode; in the case of a planner, this would be the plan itself.

outcome - some evaluation of the impact that the execution of the episode had (e.g. if a plan was successful or not)

Indexing

Episodes are stored in memory unchanged (i.e. not generalized) and are indexed for fast retrieval. We have adopted a two-layer indexing scheme similar to MAC/FAC ([Forbus *et al.*, 1995]): a *shallow indexing* in which each episode is indexed by the all its features taken in isolation and a *deep indexing* in which episodes are linked together by how they differ *structurally* from one another.

During retrieval, shallow indexing will select potentially relevant episodes from which a hill-climbing algorithm that uses semantic-matching will find the episode(s) that best match the external stimulus. A robust memory will need to employ a flexible matching algorithm, such that old situations are still recognized under new trappings. We use a semantic matcher ([Yeh *et al.*, 2003]) to compute the similarity between a new situation and a prior episode.

This semantic matcher combines subgraph isomorphism with transformation rules in order to resolve mismatches between two representations. It was successfully applied to assessing battle plans' strengths and weaknesses ([Yeh *et al.*, 2003]), questions answering [Barker *et al.*, 2004] as well as natural language tasks like building models of user utterances ([Yeh *et al.*, 2005]) and word-sense disambiguation and semantic role labeling [Yeh *et al.*, 2006]).

Memory API

The memory module provides two basic functions: **store** and **retrieve**. **Store** takes a new Episode represented as a triple [context, context, outcome] and stores the Episode in memory, indexing it along one or more dimensions; **retrieve** takes a partially specified Episode and one or more dimensions and retrieves the most similar prior episodes along those dimensions. Other information, such as how these episodes differ from the stimulus, is returned as well. This is intended to be used by the application that uses the episodic memory module in order to better make use of the returned episodes.

4 Incremental Retrieval

The general retrieval algorithm works well for memory-based planning and sentence interpretation applications, in which memory organized the patterns of entities and relations, but it will need to be modified in order to work with *sequences of events* that are *observed incrementally*.

At first glance, the fact that data is presented incrementally seems to increase retrieval time due to the need to query memory with the presentation of each new stimulus. However, incremental data reduces the size of each query. Humans are good at dealing with continuous streams of stimuli and employing expectations to focus attention and guide recognition. The question we address here is: can we devise such an algorithm for an episodic memory?

This idea has been put forth before ([Schmidt *et al.*, 1978], [Schank, 1982]) and has been applied in areas like dialogue processing ([Grosz and Sidner, 1986], [Litman and Allen, 1987]) and plan recognition ([Schmidt *et al.*, 1978]). The sequential structure of events helps constrain the type of expectations a system might form to just the next event(s) (its type and possibly its description).

To be able to take advantage of this, a memory should have the ability to ([Schmidt *et al.*, 1978]):

form hypotheses based on a set of initial observations and background knowledge

build expectations about next actions based on current hypotheses

recognize whether expectations were met when new observations become available

refine and revise a set of hypotheses when expectations are not met; this includes dropping hypotheses that don't conform to the observed stimuli and building new ones that do.

We have implemented a simple incremental retrieval algorithm:

```
initialize hypothesis
while there are stimuli do
  make a prediction about next event
  observe next event
  compare expectation with observation
  refine/revise hypotheses
```

After a new stimulus is observed, memory will refine or revise its prior hypotheses. These hypotheses consist of prior episodes that agree with the data observed so far.

5 Pattern Capture and Recognition using an Episodic Memory

The goal of this project is to augment a generic memory module for events with an retrieval algorithm that can deal with incrementally available data. We want the system to be able to make predictions incrementally, in a fast and accurate manner, comparable in performance to the state-of-art in statistical recognition. Due to the generic nature of the memory module and of the retrieval algorithm, this approach should be easily portable to new domains.

5.1 The Linux Corpus

We chose to evaluate our approach on a goal schema recognition task in the Linux Plan Corpus [Blaylock and Allen, 2004]. In goal schema recognition the purpose is to predict the type of goal an agent has, but not its exact parameters.

Plan sessions	457
Goal schemas	19
Action schemas	48
Avg. Actions/Plan	6.1

Figure 1: The Linux plan corpus statistics

This corpora is similar to the Unix corpus [Lesh, 1998], but is an order of magnitude larger in size. It was gathered from human Linux users from the University of Rochester, Department of Computer Science. Users were given a goal like `find a file with 'exe' extension` and were instructed to achieve it using simple Linux commands (no pipes, no `awk`, etc.) All user commands along with their results were recorded. For each goal, users were also asked to assess whether they accomplished it. The users judged 457 sessions to be successful[1], involving 19 goal schemas and 48 action schemas (i.e. Linux commands).

5.2 Experiment

We adapted the generic episodic memory module to work with the incremental retrieval algorithm and applied it to the goal schema recognition task on the Linux plan corpus. A 10 fold cross-validation was performed on the 457 plans, measuring the accuracy of the memory recognition algorithm and the number of events matched per recognition problem.

Plan sessions are presented to the system incrementally. After each new plan action is seen the system refines or revises its current set of hypotheses.

Due to the noise in the dataset, the recognition algorithm has to deal with superfluous actions (i.e. actions that do not influence the outcome of the plan). It does this by allowing mismatches between a new plan and prior episodes.

5.3 Example

Suppose a new plan is being observed one event at a time. Each Linux command is preceeded by `sh>` and the result of its execution appears below.[2] The system's response is given after each such command. Prior episodes are described in terms of their sequence of events.

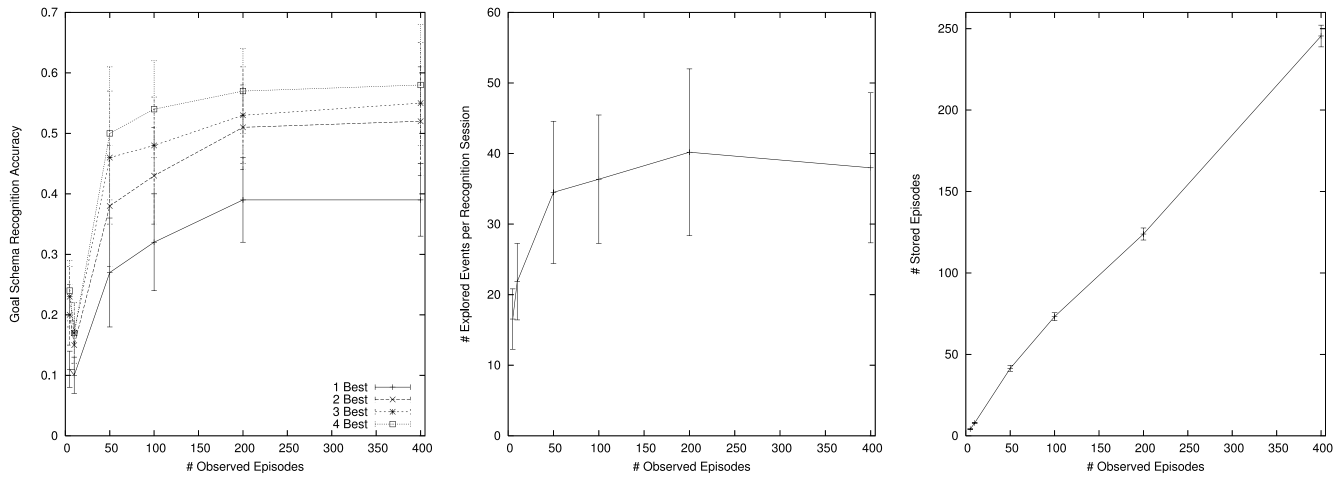
After each event is observed, memory will try to recall prior episodes that had such an event and will compare the current observation with those recalled. As a result, the set of plausible hypotheses is revised (e.g. by adding newly reminded episodes that match observation). To maintain scalability, we limit the number of remindings that the system explores (in this case to 5).

```
sh> find -name linux
./code/linux
```

Given the observed event, memory is reminded of 3 episodes:

[1]Because some users were not able correctly judge this, there are still a number of failed sessions and, therefore, data is noisy.

[2]Please note that the results are displayed here just for the readability, the system does not see them



(a) Accuracy. (b) Number of Explored Episodes. (c) Number of Stored Episodes.

Figure 2: Experimental results for the Linux planning corpus: Accuracy of goal schema recognition using Top 1, 2, 3 and 4 predictions, number of explored events per recognition and number of stored episodes.

Episode16 and **Episode14**, both consisting of a single event `Find-By-Name` and having as goal `find-file-by-attr-name-exact(b.c)`
Episode7 with events: `Find-By-Name, Ls, Tar-Zip-Create, Tar-Zip-Create, Ls` with goal `compress-dirs-by-loc-dir`.

The observed event is matched against all reminded episodes. **Episode14** and **Episode16** are removed due to mismatches.[3] **Episode7** is kept as the only plausible hypothesis.

The next event is observed:

```
sh> find -name *.pl
./code/accessor.pl
./code/constructor.pl
./code/gang/dwarf/aml.pl
```

Reminded Episodes: **Episode10**, whose goal schema is `move-files-by-attr-name-ext` and has the follosing events: `Find-By-Name, Find-By-Name, Move, Move, Move, Find-By-Name`. Since **Episode10** has not been compared to previously observed events, this comparison is done now. Although no event in **Episode7** matches the observed one, it is kept in the set of plausible hypotheses.

The last event is observed:

```
sh> mv ./bin/gang/set/convert.pl
     ./code/accessor.pl
     ./code/constructor.pl
     ./code/gang/dwarf/aml.pl
     ./code/linux
```

Reminded Episodes: **Episode10**, already among plausible ones, is matched against observed event.

[3]In this case the event type was the same, hence the reminding, but due to incorrect command syntax its parameters were missing. We tried to get a faithful translation of the original dataset and translated incorrect syntax commands, which resulted in events with no parameters.

There are no more observations and the most similar prior plan is that of **Episode10**. The system will conclude that its goal schema `move-files-by-attr-name-ext` is the most plausible one for the observed plan, which is correct.

Even this simple example shows some of the variability found in this data: files were moved individually to the desired locations in both episodes, but one used a single `Move` command, while the other moved them all together using their names. Another way to achieve the same goal is to use a regular expression as an argument for `Move`.

5.4 Experimental Results

We measured the accuracy of the top N predictions on a goal schema recognition task (Figure 2(a)). After memory is trained on 400 plans, accuracy is around 37%. This is not significantly different than previous studies on the same dataset and task ([Blaylock and Allen, 2004]), but still rather low. This is due to several facts: the data is noisy (e.g. bad commands are used - `fgrep` is used instead of `find`, user reports success although the given goal was not achieved) and some goal schemas are similar enough so that the system confuses them (e.g. `find file by extension` and `find file by name`).

Given our goal to build a generic memory module, we are also interested in the performance of memory alone, besides that on the goal recognition task. We measured the number of events per plan session that memory tried to match during recognition. This should be an indicator of retrieval cost. We also measured memory growth with the number of observed episodes. This is directly related to the storage policy. For this experiment we have implemented a simple storage policy: keep only those plans that could not be correctly recognized at the moment they were observed.

Memory grows linearly in the number of observed plans (Figure 2(c)). This is due to accuracy being around 40%, which prompts memory to keep on learning (i.e. storing plans). However, the number of explored episodes (Figure 2(b)) grows at a different rate: asymptotically fast until 200 plans have been observed and remains constant after that. In 200 to 400 observed plans interval memory size almost doubles but the number of explored episodes remains the same. This is empirical evidence that memory retrieval is scalable.

After 400 plans have been observed, around 41 actions are examined per recognition. Given that there are an average of 6.1 actions per plan, this means that an average of 6.72 plans are searched for each recognition.

5.5 Limitations of Current Approach

Our current shallow indexing scheme treats all features equally, without assigning any weight to them. As we limit the number of explored remindings, it is important that remindings generated by more distinctive features be explored before those generated by less distinctive ones.

One issue that we did not address here is the sensitivity of memory retrieval performance to noise. Even though the Linux plan corpus is noisy we don't have good measure of how much noise there is (e.g. how many sessions are misclassified as successes at accomplishing their goal; how many unnecessary actions were taken by users given a goal). We would also like to study how memory performance degrades when noise is introduced.

Another important problem not addressed here is the 'keyhole recognition problems' ([Cohen *et al.*, 1982]) in which the entity executing the plan specifically tries to hide their intentions, making the recognition task harder.

6 Future Work

Being able to make accurate predictions early on during the unfolding of the plan is very important. These predcitions constitute early warnings of potential outcomes that the user might act upon. A measure of how good the recognizer is at predicting/recognizing the plan when only a few actions have been observed is the *convergence point* - after how many observed actions the system starts making the correct prediction. We plan to measure this in a subsequent study.

Goal recognition is just one part of the picture. To be useful, it needs to be paired with parameter recognition. We plan to address this in the near future.

Our current measure of similarity of two plans is based on the similarity of their individual events. A more powerful and accurate measure needs to take into account whether/how individual actions change the state of the world.

Complex plans happen over longer periods of time and consist of many low-level events. They are unlikely to be recognized just by looking at these individual events. A good recognizer needs to be able to recognize subgoals and use them in the subsequent recognition process.

The current approach builds implicit expectations in the form of candidate episodes that match the events observed so far. A way to improve retrieval speed is for memory to actively look for how these candidate episodes differ from each other and test for the presence/absence of those differences in the new stimuli.

Expectations built by a generic episodic memory could be used to focus processing on confirming/disproving a smaller set of candidate hypotheses, by giving the application the choice of specifying the ordering function. For example, in a domain like crime prevention, one might want to test first the hypotheses that have the worst outcome, so that preventive measures could be taken as quickly as possible.

7 Conclusions

In this paper we have proposed an approach to pattern capture and recognition based on a generic memory module for events. We have showed that such a memory module can be augmented with a simple incremental retrieval algorithm in order to handle incrementally available data and to make predictions at each step. We evaluated this approach on a goal schema recognition task on a complex and noisy dataset and showed that it achieved the same level performance as state-of-art statistical approaches. Memory organization and retrieval proved scalable. Due to the generic nature of the memory module and of the retrieval algorithm, this approach should be easily portable to new domains.

References

[Barker *et al.*, 2004] K. Barker, S. Chaw, J. Fan, B. Porter, D. Tecuci, Peter Z. Yeh, V. Chaudhri, D. Israel, S. Mishra, P. Romero, and P. Clark. A question-answering system for ap chemistry: Assessing kr&r technologies. In *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 488–497, 2004.

[Blaylock and Allen, 2004] N. Blaylock and J. Allen. Statistical goal parameter recognition. In *The 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, 2004.

[Cohen *et al.*, 1982] Philip R. Cohen, C. Raymond Perrault, and James F. Allen. Beyond question answering. In *Strategies for Natural Language Processing*, pages 245–274. Lawrence Erlbaum Associates, 1982.

[Forbus *et al.*, 1995] Kenneth Forbus, Dedre Gentner, and Kenneth Law. Mac/fac: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141–205, 1995.

[Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 3(12):175–204, 1986.

[Kautz, 1997] Henry A. Kautz. A theory of plan recognition and its implementation. In J. F. Allen, H.A. Kautz, and R.N. Pelavin, editors, *Reasoning about Plans*, chapter 2. Morgan Kaufmann, San Mateo, CA, 1997. http://www.cs.washington.edu/homes/kautz/papers/prbook.ps.

[Kolodner, 1984] Janet L. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.

[Könik and Laird, 2006] Tolga Könik and John E. Laird. Learning goal hierarchies from structured observations and expert annotations. *Machine Learning*, (in print), 2006.

[Lesh, 1998] N. Lesh. *Scalable and Adaptive Goal Recognition*. PhD thesis, University of Washington, 1998.

[Litman and Allen, 1987] Diane Litman and James Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200, 1987.

[Nuxoll and Laird, 2004] Andrew Nuxoll and John E. Laird. A cognitive model of episodic memory integrated with a general cognitive architecture. In *Proceedings of the Sixth International Conference on Cognitive Modeling*, pages 220–225, Mahwah, NJ, 2004. Lawrence Earlbaum.

[Pollack, 1990] Martha Pollack. Plans as complex mental attitudes. In Philip R. Cohen, Jerry Morgan, and Martha Pollack, editors, *Intentions in Communication*, pages 77–103. MIT Press, Cambridge, Massachusetts, 1990.

[Porter *et al.*, 1990] Bruce W. Porter, Ray Bareiss, and Robert C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artifical Intelligence*, 45:229–263, 1990.

[Ram and Santamaria, 1997] Ashwin Ram and J. C. Santamaria. Continuous case-based reasoning. *Artificial Intelligence*, 90(1-2):25–77, 1997.

[Schank, 1982] Roger C. Schank. *Dynamic Memory. A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.

[Schmidt *et al.*, 1978] Charles F. Schmidt, N. S. Sridharan, and J. L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11:45–83, 1978.

[Tulving, 1983] Endel Tulving. *Elements of Episodic Memory*. Clarendon Press, Oxford, 1983.

[Vere and Bickmore, 1990] Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*, 4:41–60, 1990.

[Woods, 1990] W. A. Woods. On plans and plan recognition: Comments on pollack and on kautz. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 135–140. MIT Press, Cambridge, MA, 1990.

[Yeh *et al.*, 2003] Peter Z. Yeh, Bruce Porter, and Ken Barker. Using transformations to improve semantic matching. In *K-CAP 2003: Proceedings of the Second International Conference on Knowledge Capture*, pages 180–189, 2003.

[Yeh *et al.*, 2005] Peter Z. Yeh, Bruce Porter, and Ken Barker. Matching utterances to rich knowledge structures to acquire a model of the speaker's goal. In *K-CAP2005: Proceedings of Third International Conference on Knowledge Capture*, pages 129–136, 2005.

[Yeh *et al.*, 2006] Peter Z. Yeh, B. Porter, and K. Barker. A unified knowledge based approach for sense disambiguation and semantic role labeling. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006)*, 2006.