Reinforcement Learning-based Feature Selection For Developing Pedagogically Effective Tutorial Dialogue Tactics

Min Chi, Pamela Jordan, Kurt VanLehn, Moses Hall
{mic31, pjordan+, vanlehn+, mosesh}@pitt.edu
Learning Research Development Center & Intelligent System Program,
University of Pittsburgh

Abstract. Given the subtlety of tutorial tactics, identifying effective pedagogical tactical rules from human tutoring dialogues and implementing them for dialogue tutoring systems is not trivial. In this work, we used reinforcement learning (RL) to automatically derive pedagogical tutoring dialog tactics. Past research has shown that the choice of the features significantly affects the effectiveness of the learned tactics. We defined a total of 18 features which we classified into four types. First, we compared five feature selection methods and overall upper-bound method seems to be most efficient. Then we compared the four types of features and found that temporal situation and autonomy related features are significantly more relevant and effective to tutorial decisions than either performance or situation related features.

1 Introduction

One challenging issue confronting dialogue tutoring systems is how to select the best tutoring actions for any student state so as to maximize learning gains. We call this decision making *tutoring tactics* since it generally governs brief episodes of tutoring dialogue, such as a single step, and seems to be crucial for achieving further improvements in pedagogical effectiveness. Bad tutorial decisions have the potential to bore or frustrate students to the point that they fail to learn. Many existing dialogue systems use hand-crafted tactical rules derived from analyzing human dialogues [6,13]. However, identifying and implementing these tactical rules is not trivial. It has been shown that expert human tutors generally employ a range of tutoring actions and their decisions depend in subtle ways on the student's competence, student's self-confidence and other factors. For example, one important tutoring tactic is whether the tutor should *tell* the student the answer directly for a step or whether he should *elicit* the step from the student with a prompt or a series of questions. Collins et al. suggest that when a student is unfamiliar with the target knowledge, the tutor should tell it directly; when a student becomes more familiar with the knowledge, the tutor should elicit the answer via questions; and once a student has mastered the knowledge, it doesn't matter whether it is told or elicited [4,5].

Most existing dialogue tutoring systems with hand-crafted rules react sensibly when students exhibit less than optimal behaviors; however, their tutorial interactions are much stiffer than those of human tutors and they have not yet achieved the same effectiveness that one-on-one, face-to-face expert human tutors have [1]. In recent years, work in designing spoken dialogue systems has proposed several data-driven methodologies; one of them is Reinforcement Learning (RL) [3,12,13]. It has been shown that RL is effective at automatically learning the best action to take at any state in a dialogue. And such success inspired us to use RL for dialogue tutoring systems.

Our work can be divided into three stages. In the first stage, we built an initial tutorial dialogue system and collected an exploratory corpus by using the system to train 64 students. In this initial system, decisions on certain types of tutoring actions were made randomly. In the second stage, we used RL on the exploratory corpus to derive tutoring tactics for these actions and then incorporated the learned tactics into the initial system. The modified system is identical to the initial one except that the tutoring actions that were previously made randomly are now based on

the learned tutoring tactics. In the last stage, the modified system is being used to train a new group of students so that we compare their learning performance to that of the students previously trained on the initial system. We expect the modified system to be so well attuned to students that it will be more effective than the initial system.

One tactical decision investigated in this study was whether the tutor should *tell* the next step to the student or *elicit* it from the student. We defined 18 features to model the dialogue and students' state. They were selected based upon the four types of features that were shown in [7] to be relevant for human tutors to making their tutorial decisions: autonomy, temporal situation, situation, and performance respectively. Among 18 features, features 1-3 are autonomy related based upon the amount of the work that the tutor has let students do; features 4-8 cover time-related information such as time spent on the training so far; features 9-11 are situation related including whether the student and tutor are now problem solving or in post discussion, the difficulty level of the problem and so on; and finally, features 12-18 cover the performance information such as the correctness of student's previous answers and the ability of student.

In an RL model, the size of the state space increases exponentially as the number of involved features increases. In order to learn effective tutoring tactics, we should have a corpus that covers each of these states at least once, which means $\geq 2^{18}$ in our case. However, it is almost impossible to do so given the high cost of collecting educational data. On the other hand, the learned policy may become too subtle to be necessary. Figure 1 shows an example of a learned policy involving five features: [1, 2, 4, 15, 16], in which feature 2 is an autonomy feature defined as the percentage of elicitation the students have received so far. Each of the five features was converted from a real number to a binary value based upon their median scores, for example, f2='.4982' means if feature 2 value is above .4982, it is 1 otherwise, it is 0. There were a total of 32 rules learned: in 10 situations the tutor should elicit, in 19 it should tell; in the remaining 3 cases either will do. For example, when all of the features are 0 then the tutor should tell as 0:0:0:0:0 the first in the list of tells. As you can see, five features provide a large space is already quite complex and is already much more subtle than most of the tutorial tactics derived from analizing human turorial dialogues [4,5].

```
KC22    'features'=[1, 2, 4, 15 ,16],
        'cutoff'=[f1='1.0000' f2='.4982' f4='56.0000'  f15='.6154'    f='.2683'  ],
        'policy':
          'elicit: [0:0:0:0:1, 0:0:0:1:0, 1:0:0:0:0, 1:0:0:0:1, 1:0:0:1:0, 1:1:0:1:1,
                    0:0:1:0:0, 0:0:1:0:1, 1:0:1:0:0, 1:0:1:0:1],
          'tell:   [0:0:0:0:0, 0:0:0:1:1, 0:1:0:0:0, 0:1:0:0:1, 0:1:0:1:0, 0:1:0:1:1, 1:1:0:0:1,
                    1:1:0:1:0, 0:0:1:1:0, 0:0:1:1:1, 0:1:1:0:0, 0:1:1:0:1, 0:1:1:1:0, 0:1:1:1:1,
                    1:0:1:1:1, 1:1:1:0:0, 1:1:1:0:1, 1:1:1:1:0, 1:1:1:1:1]
           'else: [1:0:0:1:1, 1:1:0:0:0, 1:0:1:1:0]
```

Figure 1. A Learned Policy Based On Five Features On KC22

Based on the size of the exploratory corpus we collected in the first stage, we decided that no more than *six* features should be used. Previous research involving RL and feature selection in dialogue systems either focused on selecting features with certain characteristics [2,7,8] or investigated a relatively small number of features [10,11]. Therefore, in this study, we proposed four general RL-based feature selection methods.

2 Background

Past research on using RL to improve spoken dialogue systems has commonly used Markov Decision Processes (MDP's). An MDP [8,9] describes a stochastic control process and formally corresponds to a 4-tuple (S,A,T,R), where $S = \{s_i\}_{i=1..m}$, is a finite set of process states; $A=\{a_k\}_{k=1..m}$ is a finite set of actions; $T : S \times A \times S \rightarrow [0, 1]$ is a set of transition probabilities between states that describe the dynamics of the modeled system; for example: $P_t(s_i|s_j, a_k)$ is the probability that the model would transition from state s_j to state s_i by taking action a_k at time t; and $R : S \times A \times S \rightarrow R$ denotes a reward model that assigns rewards to state transitions and models payoffs associated with such transitions. The goal of using MDPs is to determine the best policy π^*: the set of actions the model should take at each state s_i to maximize its expected cumulative utility (V-value), which can be calculated from the following recursive equation:

$$V(s_i) = \sum_{s_j} P_{s_i s_j}^{\pi^*(s_i)} [R_{s_i s_j}^{\pi^*(s_i)} + \gamma V(s_j)]$$

As long as a proper state space, action space and reward function are set up, an MDP allows one to automatically derive and compute the best policy. For dialogue tutoring systems, deriving effective tutorial tactics from tutoring dialogues can be naturally cast in the MDP formalism: for the state space, each s_i can be viewed as a vector of features representing the tutoring dialogue context, the student's knowledge level and so on. Action space corresponds to tutoring actions, e.g. elicit or tell; the reward function corresponds to students' learning gains. For tutoring dialogues, the reward is a *delayed* reward because for each state the reward is not known until students have completed the tutoring and taken the post-test.

In this work, we used an MDP package that was designed for Tetreault & Litman's studies since it has proven to be both reliable and successful [10,11]. In previous studies, Tetreault & Litman primarily investigated methods for evaluating whether certain features would improve policy effectiveness. There are two main differences between our study and theirs: firstly, they used a previously collected corpus that was not exploratory with respect to tutorial tactics to train an MDP model in that the tutor often used only one type of action in many dialogue states, which severely limited the types of questions that they could investigate[10,11]; while we use a more exploratory corpus by training students on a dialogue tutoring system in which multiple actions can often be taken and random tactical decisions were made. Thus it is better suited for creating an MDP. Secondly, they did not need to address the problem of general feature selection methods since they only used five features while we had to select up to six of 18.

To evaluate the learned policies, we use three criteria: Expected Cumulative Reward (ECR) and lower-bound and higher-bound of the 95% confidence interval. ECR is the average reward one would expect in the MDP and it is calculated by normalizing the V-value of each state by the number of times it occurs as a start state in a dialogue and then summing over all states [11]. The confidence interval represents how reliable the ECR of a learned policy is [10]. The wider it is, the less reliable the policy's ECR is.

For example, for a learned policy ***A*** derived from feature 2 alone on definition of spring potential energy (KC22) is "if the percent of elicit so far is less than 49.82%, tutor should elicit otherwise the tutor should tell."; ***A*** has ECR = 3.02 (range [-100, 100]) with a 95% confidence interval= [-2.71, 8.45], which means there is a 95% chance that the ECR of the learned policy is between a lower-bound of -2.71 and an upper-bound of 8.45. In Figure 1, also on KC22 but involves five features: 1, 2, 4, 15 and 16. It has ECR = 44.29 with a 95% confidence interval= [23.49, 50.51], which is much more effective than policy ***A*** because even its lower-bound is much higher than policy A's upper-bound.

Sometimes we encounter situations in which the ECR for A is the same as for B; but the confidence interval of A is much narrower than that of B. By only using the three criteria described above, policy A and B cannot be compared. Therefore, we define the hedge of a learned policy as:

$$Hedge = \frac{ECR}{(\{upper-bound\} - \{lower-bound\})}$$

By using the hedge, we can say that policy A is better than policy B since the hedge of A is higher than the hedge of policy B.

3 Experiment

The domain chosen for this study is work and energy covered in college physics. The procedure was as follows students: (1) read a short textbook, which describes the major principles and concepts; (2) take a pre-test; (3) work through seven open-ended training problems with a dialogue tutoring system (Cordillera [14]); and (4) take a post-test that is identical to the pre-test. In the stage one, 64 college students who had not taken any college physics completed the experiment, receiving payment for their participation. Students needed 8-15 hours to complete the procedures and usually required 4-6 sessions of about 2 hours each. Thus, the collected corpus comprises 64 human-computer tutoring dialogues and each dialogue is an extended interaction with one student that covers seven different college-level physics problems. The number of state-action pairs in each of the 64 collected tutorial dialogues varies from 700 to 900.

All training problems, and all pre- and post-tests problems were selected from 127 quantitative and qualitative problems collected from various physics literature. In order to solve these 127 problems, 32 unique knowledge components (KCs) were identified as necessary. A KC is "a generalization of everyday terms like concept, principle, fact, or skill, and cognitive science terms like schema, production rule, misconception, or facet" [14]. For example, KC22[1] is both a concept and a principle while KC23[2] is a fact. KC22 consists of both procedural knowledge and declarative knowledge while KC23 is mainly declarative knowledge and thus learning these two KCs clearly involves different cognitive skills. Therefore, for different KCs, we expect that different features should be considered when making a tactical decision and different tactical decisions should be derived.

In order to learn KC-specific tutorial tactics, students' pre- and post-test scores were also calculated per KC. It turned out our training, pre- and post-problems covered 27 of the 32 possible KCs. Three of the 27 KCs showed up in the tutoring dialogue but were not associated with any action decision points; five KCs coincided only once or twice with decision points; and one KC did not appear in the pre- and post-tests. Therefore, we were left with 18 KCs for which it is possible for us to derive tutoring tactics. Comparisons of pre- and post-test scores indicated that students did learn during their training with Cordillera: their post-test scores were significantly higher than their pre-test scores: $t(126)= 3.36$, $p= 0.001$. Similar to Tetreault & Litman's work, we used students' normalized learning gains (NLGs) for each KC_i as the *delayed* reward function, which is defined as:

$$NLG(KC_i) = \frac{Posttest(KC_i) - \Pr etest(KC_i)}{1 - \Pr etest(KC_i)}.$$

[1] If an object and a spring are in a system, then their spring potential energy is 0.5* k*d^2, where k is the spring constant and d is the displacement of the object relative to the equilibrium position of the spring.
[2] The unit for energy is the Joule.

For each KC_i, the 64 students were split into two groups by the median value of their NLG on it, so the better-performing students' dialogues were given a positive reward of +100, while the other half were assigned a negative reward of -100. The rewards were assigned in the final dialogue state.

Additionally, we annotated our tutoring dialogues and action decisions based on which KCs a tutor action or tutor-student pair of turns covered; Table 1 shows an example. Here lines 1, 5, 6, 7, 9 and 11 are tutor turns, the action type and their target KCs are listed on the right. For example, in the first utterance is an elicit action regarding KC22 while the fifth is a tell action directly stating KC22. Lines 2 and 12 are student responses and they are both correct (as indicated by the '+'); row 2 is about KC22 while row 12 is about KC27.

Table 1. Examples of Tutorial Dialogue with KC Labels

1.	T: At T2 we also know the spring is uncompressed. So, what type of energy can we easily find now?	elicit, KC22
2.	S: spe	+KC22
3.	T: Yes, I agree.	
4.	T: At this point we know the values for KE2 and GPE2. It is time to compute the value for SPE2.	
5.	T: It is important to remember that we cannot find the SPE of an OBJECT. We need to find the spring potential energy of a pair: object-spring.	tell, KC22
6.	T: It is possible to compute the spring potential energy at T2 because we know the spring is uncompressed. That is, d2=0 m.	tell
7.	T: Using this information we can infer that the spring potential energy of the block-spring pair at T2 is SPE2=0 J.	tell, KC22 KC23
8.	T: Now we have all we need to…	
9.	T: TME of the system at T2 is the sum of the block`s KE and the block-Earth pair`s GPE because the value for the spring potential energy is zero. In this case TME2=11.88 J+1.62 J =13.5 J.	tell, KC23 KC24
10.	T: At this point we have found the value for TME2. We can now easily find the system`s TME at all other time points since the system is isolated.	
11.	T: What principle gives us the system`s total mechanical energy at every time point, knowing its value at T2 and also given that the system is isolated?	elicit, KC27
12.	S: student conservation of energy	+ KC27
13.	T: Yes.	

4 Methods

The four feature selection approaches we propose in this paper are fairly straightforward and share the same procedure, which consists of the three phases described below.

Phase 1: For each of 18 features in the state space, use MDP to get a single-feature-policy.
Phase 2: Sort all of the features based on the learned single-feature-policies from ***high to low*** by one of the following measures:
- ECR
- Lower-bound
- Upper-bound
- Hedge

Phase 3: Start with the first feature in the sorted feature list, add one feature at a time to the MDP and learn a new policy. Repeat this process repeat 5 times.

Based on the sorting criteria in phase 2, we named our four feature selection methods: ECR, lower-bound, upper-bound, and hedge respectively. Since these values were calculated from the single-feature-policies learned using MDP in phase 1, the four methods are RL-based. We expect them to be more effective than a random selection. To test our expectations, we created 120 random policies by running the MDP package on randomly selected features (20 rounds for each randomly selected feature set, where each set contained between one and six features). Therefore, for each KC and each feature set size, we learned one tutoring tactic for each RL-based method plus 20 for random feature selection. This gave us (1*4+20)*6 =144 policies.

For each KC_i, we selected one policy that had the highest ECR, lower-bound and upper-bound from the 144 learned policies and named it the "best policy". To quantitatively evaluate how much less effective a learned tutoring tactic is than the best policy, we defined a normalized ECR (NECR) for a learned tutoring tactic as:

$$\mathrm{NECR}(\mathrm{KC_i}, \mathrm{N}, \mathrm{Method_k}) = \frac{1}{C}\sum_{C}\frac{\mathrm{ECR}(\mathrm{KC_i}, \mathrm{N}, \mathrm{Method_k}) - \mathrm{Min\_ECR}(\mathrm{KC_i})}{\mathrm{Max\_ECR}(\mathrm{KC_i}) - \mathrm{Min\_ECR}(\mathrm{KC_i})}$$

$$Max\_ECR(KC_i) = \max_{N\in\{1..6\}, me\{all\ methods\}}(ECR(KC_i, N, Method_m)$$

$$Min\_ECR(KC_{ij}) = \min_{N\in\{1..6\}, me\{all\ methods\}}(ECR(KC_i, N, Method_m)$$

$\mathrm{Max\_ECR}(\mathrm{KC_i})$ and $\mathrm{Min\_ECR}(\mathrm{KC_i})$ are the maximum and minimum ECR among all 144 of the learned policies for KC_i. C is a constant with C=1 for each of the four RL-based methods and C=20 for random feature selection. The maximum NECR for a learned policy is 1 if the learned policy is the best one and the minimum is 0 if the learned policy is the worst one.

5 Results

In order to evaluate the feature selection methods and how the feature effectiveness, we have two main goals. First, we will compare the four RL-based feature selection methods against random feature selection; second we will investigate which features seem to be more important for deriving the best tutoring tactics for deciding to elicit/tell across all KCs.

5.1 Comparing four RL-based methods against random feature selection

Table 2. Comparing the Average NECR of Five Selection Methods for Increasing Feature Set Sizes

Number Features Involved	Upper-bound	ECR	Hedge	Lower-bound	Random
1	0.345	0.372	0.337	0.355	0.119
2	0.355	0.370	0.335	0.314	0.196
3	0.416	0.447	0.400	0.352	0.275
4	0.550	0.515	0.520	0.419	0.348
5	**0.682**	0.579	0.573	0.435	0.422
6	0.673	0.614	0.594	0.485	0.480

NECR is defined as how much a learned tutoring tactics is less effective than the best policy. Table 2 shows the average NECR given the number of selected features and feature selection

method across all 18 KCs. As expected, random feature selection has the worst average NECR regardless of the number of features involved. Overall, if the number of features ≤3, then the ECR approach is the best feature selection method; but if the number of features is between 4 and 6, then the upper-bound method is the best. As the number of features involved increases, the effectiveness of the learned policies for elicit/tell tends to become better, except for upper-bound, which has a better average NECR when there are five features than when there are six. Overall, using the upper-bound method to select five features seems to be the most effective method across all KCs on elicit/tell action decision.

On the other hand, across all 18 KCs the number of times that each method found the best policy is: fourteen times for random, three for upper-bound, two for ECR, one for hedge and none for lower-bound method. The total did not add up to 18 KCs because for KC1, three methods, ecr, hedge, and upper-bound all found the same best policy. Therefore, although random feature selection has the worse average NECR than other four RL-based methods, it is an effective way to find best policies. However, note that the random feature selection method was repeated 120 times for each KC and so it has a total of 120*18=2160 chances to find the 14 best policies; while the upper-bound method was applied only 6 times for each KC and thus has a total of 108 chances. Additionally, because our state space is still relatively small, we expect that the performance of random selection would decrease significantly as the number of features in the state space increases. However, for the four RL-based feature selection methods, increasing the number of features would not decrease their effectiveness since they do not directly depend on the number of features in the state space.

Moreover we compared the best policy learned for each KC by the upper-bound method and those learned by random selection. Over all 18 KCs, the upper-bound policies with just 108 attempts were only 9.46% less effective than random feature selection policies with 2160 attempts. These results indicate that that in education, features that may result higher learning gains should always be always be considered by the tutor when making decisions. This is likely due to the fact that in the worst case a student will simply not learn rather than lose information so the cost of incorporating superfluous feature is low..

5.2 Frequency of Features in Best Policies

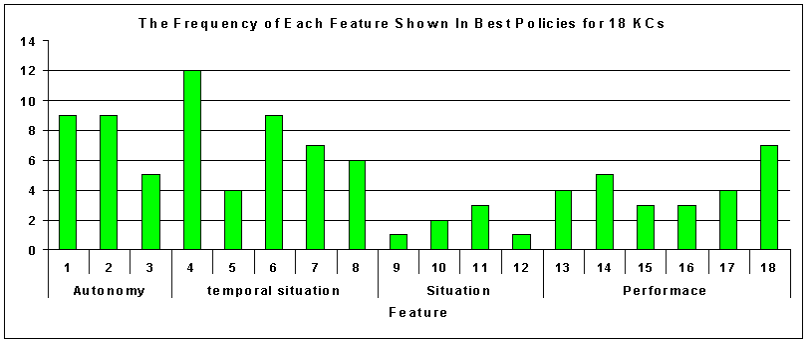

Figure 2. The Frequency of Each Feature Shown In Best Policies.

Figure 2 shows the frequency with which each feature appears in the best policies. This frequency differs significantly among the four feature types: $F(3)= 7.47$, $p=0.003$. There is no significant difference between the three autonomy and five temporal situation related features. When combined they are significantly more frequent than either the performance or situation related features: $t(12)= 2.74$, $p=0.18$ and $t(10)= 4.26$, $p=0.002$ respectively. Consistent with previous research based on analyzing human tutorial dialogue, autonomy related features seemed to be more relevant in deriving effective tutorial tactics. Additionally, we found that temporal

situation related features were also relevant, even more so than the performance related ones when deciding whether to elicit or tell. This was not indicated in previous literature on human tutorial dialog analysis. One possible explanation for this was that in most of the prior literature, temporal situation related factors were often not considered.

6 Conclusions and Future Work

In this paper, we described our work on applying RL methods to derive effective tutoring tactics for elicit/tell. We showed that deriving effective tutoring tactics from tutoring dialogues can be cast in the MDP formalism. Additionally, we proposed four RL-based domain-general feature selection methods. We found the upper-bound method to be more effective than the others.

One of our goals for future work is to investigate how to still get reasonable policies without annotating individual KCs in the dialogues. The annotation process is prohibitively time-consuming and it is not unusual for domain experts to disagree [14]. Another of our goals for future work is to determine how to learn one reasonable policy for all KCs without sacrificing too much of the expected effectiveness. Another two important issues are how to avoid the expensive initial data collection and how to combine the new data with the existing data so that we can learn even more powerful policies.

Acknowledgments
We would like to thank the NLT group and Diane J. Litman for their comments. Support for this research was provided by NSF grants #0325054.

References

[1] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring.," Educational Researcher, 1984. no. 13, p. 4–16,
[2] M. Frampton and O. Lemon. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *IJCAI Wkshp. on K&R in Practical Dialogue Systems*, 2005.
[3] J. Henderson, O. Lemon, and K. Georgila. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In IJCAI Wkshp. on K&R in Practical Dialogue Systems, 2005.
[4] Collins, A., and Stevens, A Goals and Strategies of Inquiry Teachers: In R. Vlaser (Ed.) Advances in Instructional Psychology, 1982. vol. 2 Hillsdale, NJ: Erlbaum Assoc. p. 65-119.
[5] Collins, A.. Design issues for learning environments. In S. Vosniadou, E. D. Corte, R. Glaser, & H. Mandl (Eds.), International perspectives on the design of technology-supported learning environments, 1996. p. 347-361. Mahwah, NJ: Erlbaum,
[6] M. Evens and J. Michael, One-on-One Tutoring by Humans and Computers, Lawrence Erlbaum Associates, Inc., 2006.
[7] Johanna D. Moore, Kaska Porayska-Pomsta, Sebastian Varges, Claus Zinn: Generating Tutorial Feedback with Affect. FLAIRS Conference, 2004.
[8] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement learning for spoken dialogue systems. In *Proc. NIPS '99*, 1999.
[9] R. Sutton and A. Barto. *Reinforcement Learning*. The MIT Press, 1998.
[10] J. Tetreault, D. Bohus, and D. Litman, "Estimating the reliability of mdp policies: a confidence interval ap proach," in *NAACL*, 2007.
[11] J. Tetreault and D. Litman. Comparing the utility of state features in spoken dialogue using reinforcement learning. In *NAACL*, 2006.
[12] J. Williams, P. Poupart, and S. Young. Factored partially observable markov decision processes for dialogue management. In IJCAI Wkshp. on K&R in Practical Dialogue Systems, 2005.
[13] M. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. JAIR, 12, 2000.
[14] VanLehn, K, Jordan, & Diane. Developing pedagogically effective tutorial dialogue tactics: Experiments and a testbed. In proceedings of SLaTE Workshop, 2007.