Event Extraction in a Plot Advice Agent

Harry Halpin
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
Scotland, UK
H.Halpin@ed.ac.uk

Johanna D. Moore
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
Scotland, UK
J.Moore@ed.ac.uk

Abstract

In this paper we present how the automatic extraction of events from text can be used to both classify narrative texts according to plot quality and produce advice in an interactive learning environment intended to help students with story writing. We focus on the story rewriting task, in which an exemplar story is read to the students and the students rewrite the story in their own words. The system automatically extracts events from the raw text, formalized as a sequence of temporally ordered predicate-arguments. These events are given to a machine-learner that produces a coarse-grained rating of the story. The results of the machine-learner and the extracted events are then used to generate fine-grained advice for the students.

1 Introduction

In this paper we investigate how features of a text discovered via automatic event extraction can be used in both natural language understanding and advice generation in the domain of narrative instruction. The background application is a fully automated plot analysis agent to improve the writing of students could be used by current narrative tutoring systems (Robertson and Wiemer-Hastings, 2002). As shown by participatory design studies, teachers are interested in a plot analysis agent that can give online natural language advice and many students enjoy feedback from an automated agent (Robertson and Cross, 2003). We use automatic event extraction to create a story-independent automated agent that can both analyze the plot of a story and generate appropriate advice.

1.1 The Story Rewriting Task

A task used in schools is the *story rewriting task*, where a story, the *exemplar story*, is read to the students, and afterwards the story is rewritten by each student, providing a corpus of *rewritten stories*. This task tests the students ability to both listen and write, while removing from the student the cognitive load needed to generate a new plot. This task is reminiscent of the well-known "War of the Ghosts" experiment used in psychology for studying memory (Bartlett, 1932) and related to work in fields such as summarization (Lemaire et al., 2005) and narration (Halpin et al., 2004).

1.2 Agent Design

The goal of the agent is to classify each of the *rewritten stories* for overall plot quality. This rating can be used to give "coarse-grained" *general advice*. The agent should then provide "fine-grained" *specific advice* to the student on how their plot could be improved. The agent should be able to detect if the story should be re-read or a human teacher summoned to help the student.

To accomplish this task, we extract events that represent the entities and their actions in the plot from both the exemplar and the rewritten stories. A plot comparison algorithm checks for the presence or absence of events from the exemplar story in each rewritten story. The results of this algorithm will be used by a machine-learner to classify each story for overall plot quality and provide general "canned" advice to the student. The features statistically shared by "excellent" stories represent the important events of the exemplar story. The results of a search for these important events in a rewritten story provides the input needed by templates to generate specific advice for a student.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 857–864,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2 Corpus

In order to train our agent, we collected a corpus of 290 stories from primary schools based on two different exemplar stories. The first is an episode of "The Wonderful Adventures of Nils" by Selma Lagerloff (160 stories) and the second a re-telling of "The Treasure Thief" by Herodotus (130 stories). These will be referred to as the "Adventure" and "Thief" corpora.

2.1 Rating

An experienced teacher, Rater *A*, designed a rating scheme equivalent to those used in schools. The scheme rates the stories as follows:

1. *Excellent*: An excellent story shows that the student has "read beyond the lines" and demonstrates a deep understanding of the story, using inference to grasp points that may not have been explicit in the story. The student should be able to retrieve *all* the important links, and not all the details, but the right details.

2. *Good*: A good story shows that the student understood the story and has "read between the lines." The student recalls the main events and links in the plot. However, the student shows no deep understanding of the plot and does not make use of inference. This can often be detected by the student leaving out an important link or emphasizing the wrong details.

3. *Fair*: A fair story shows that student has listened to the story but not understood the story, and so is only trying to repeat what they have heard. This is shown by the fact that the fair story is missing multiple important links in the story, including a possibly vital part of the story.

4. *Poor*: A poor story shows the student has had trouble listening to the story. The poor story is missing a substantial amount of the plot, with characters left out and events confused. The student has trouble connecting the parts of the story.

To check the reliability of the rating scheme, two other teachers (Rater B and Rater C) rated subsets (82 and 68 respectively) of each of the corpora. While their absolute agreement with Rater A makes the task appear subjective (58% for B and 53% for C), their relative agreement was high, as almost all disagreements were by one level in the rating scheme. Therefore we use Cronbach's α and τ_b instead of Cohen's or Fleiss' κ to take into account the fact that our scale is ordinal. Between Rater A and B there was a Cronbach's α statistic of .90 and a Kendall's τ_b statistic of .74. Between Rater B and C there was a Cronbach's α statistic of .87 and Kendall's τ_b statistic of .67. These statistics show the rating scheme to be reliable and the distribution of plot ratings are given in Table 1.

Class	Adventure	Thief
1 (Excellent)	.231	.146
2 (Good)	.300	.377
3 (Fair)	.156	.292
4 (Poor)	.313	.185

Table 1: **Probability Distribution of Ratings**

2.2 Linguistic Issues

One challenge facing this task is the ungrammatical and highly irregular text produced by the students. Many stories consist of one long run-on sentence. This leads a traditional parsing system with a direct mapping from the parse tree to a semantic representation to fail to achieve a parse on 35% percent of the stories, and as such could not be used (Bos et al., 2004). The stories exhibit frequent use of reported speech and the switching from first-person to third-person within a single sentence. Lastly, the use of incorrect spelling e.g., "stalk" for "stork" appearing in multiple stories in the corpus, the consistent usage of homonyms such as "there" for "their," and the invention of words ("torlix"), all prove to be frequent.

3 Plot Analysis

To automatically rate student writing many tutoring systems use Latent Semantic Analysis, a variation on the "bag-of-words" technique that uses dimensionality reduction (Graesser et al., 2000). We hypothesize that better results can be achieved using a "representational" account that explicitly represents each event in the plot. These semantic relationships are important in stories, e.g., "The thief jumped on the donkey" being distinctly different from "The donkey jumped on the thief." What characters participate in an action matter, since "The king stole the treasure" reveals a major

misunderstanding while "The thief stole the treasure" shows a correct interpretation by the student.

3.1 Stories as Events

We represent a story as a sequence of *events*, $p_1...p_h$, represented as a list of predicate-arguments, similar to the event calculus (Mueller, 2003). Our predicate-argument structure is a minimal subset of first-order logic (no quantifiers), and so is compatible with case-frame and dependency representations. Every *event* has a *predicate* (function) p that has one or more *arguments*, $n_1...n_a$. In the tradition of Discourse Representation Theory (Kamp and Reyle, 1993), our current predicate argument structure could be converted automatically to first order logic by using a default existential quantification over the predicates and joining them conjunctively. Predicate names are often verbs, while their arguments are usually, although not exclusively, nouns or adjectives. When describing a set of events in the story, a superscript is used to keep the arguments in an event distinct, as n_5^2 is argument 2 in event 5. The same argument name may appear in multiple events. The plot of any given story is formalized as an *event structure* composed of h events in a partial order, with the partial order denoting their temporal order:

$$p_1(n_1^1, n_1^2, ...n_1^a), ..., p_h(n_h^2, n_h^4...n_h^c)$$

An example from the "Thief" exemplar story is "The Queen nagged the king to build a treasure chamber. The king decided to have a treasure chamber." This can be represented by an event structure as:

$nag(king, queen)$
$build(chamber)$
$decide(king)$
$have(chamber)$

Note due the ungrammatical corpus we cannot at this time extract neo-Davidsonian events. A sentence maps onto one, multiple, or no events. A unique name and closed-world assumption is enforced, although for purposes of comparing event we compare membership of argument and predicate names in WordNet synsets in addition to exact name matches (Fellbaum, 1998).

4 Extracting Events

Paralleling work in summarization, it is hypothesized that the quality of a rewritten story can be defined by the presence or absence of "semantic content units" that are crucial details of the text that may have a variety of syntactic forms (Nenkova and Passonneau, 2004). We further hypothesize these can be found in chunks of the text automatically identified by a chunker, and we can represent these units as predicate-arguments in our event structure. The event structure of each story is automatically extracted using an XML-based pipeline composed of NLP processing modules, and unlike other story systems, extract full events instead of filling in a frame of a story script (Riloff, 1999). Using the latest version of the Language Technology Text Tokenization Toolkit (Grover et al., 2000), words are tokenized and sentence boundaries detected. Words are given part-of-speech tags by a maximum entropy tagger from the toolkit. We do not attempt to obtain a full parse of the sentence due to the highly irregular nature of the sentences. Pronouns are resolved using a rule-based reimplementation of the CogNIAC algorithm (Baldwin, 1997) and sentences are lemmatized and chunked using the Cass Chunker (Abney, 1995). It was felt the chunking method would be the only feasible way to retrieve portions of the sentences that may contain complete "semantic content units" from the ungrammatical and irregular text. The application of a series of rules, mainly mapping verbs to predicate names and nouns to arguments, to the results of the chunker produces events from chunks as described in our previous work (McNeill et al., 2006). The accuracy of our rule-set was developed by using the grammatical exemplar stories as a testbed, and a blind judge found they produced 68% interpretable or "sensible" events given the ungrammatical text. Students usually use the present or past tense exclusively throughout the story and events are usually presented in order of occurrence. An inspection of our corpus showed 3% of stories in our corpus seemed to get the order of events wrong (Hickmann, 2003).

4.1 Comparing Stories

Since the student is rewriting the story using their own words, a certain variance from the plot of the exemplar story should be expected and even rewarded. Extra statements that may be true, but are not explicitly stated in the story, can be inferred by the students. Statements that are true but are not highly relevant to the course of the

plot can likewise be left out. Word similarity must be taken into account, so that "The king is protecting his gold" can be recognized as "The pharaoh guarded the treasure." Characters change in context, as one character that is described as the "younger brother" is from the viewpoint of his mother "the younger son." So, building a model from the events of two stories and simply checking equivalence can not be used for comparison, since a wide variety of partial equivalence must be taken into account.

Instead of using absolute measures of equivalence based on model checking or measures based on word distribution, we compare each story on the basis of the presence or absence of events. This approach takes advantage of WordNet to define synonym matching and uses the relational structure of the events to allow partial matching of predicate functions and arguments. The events of the exemplar story are assumed to be correct, and they are searched for in the rewritten story in the order in which they occur in the exemplar. If an event is matched (including using WordNet), then in turn each of the arguments attempts to be matched.

This algorithm is given more formally in Figure 1. The complete event structure from the exemplar story, E, and the complete event structure from the rewritten story R, with each individual event predicate name labelled as e and r respectively, and their arguments labelled as n in either N_e and N_r. *SYN(x)* is the synset of the term x, including hypernyms and hyponyms except upper ontology ones. The results of the algorithm are stored in binary vector F with index i. 1 denotes an exact match or WordNet synset match, and 0 a failure to find any match.

4.2 Results

As a baseline system LSA produces a similarity score for each rewritten story by comparing it to the exemplar, this score is used as a distance metric for a k-Nearest Neighbor classifier (Deerwester et al., 1990). The parameters for LSA were empirically determined to be a dimensionality of 200 over the semantic space given by the recommended reading list for American 6th graders (Landauer and Dumais, 1997). These parameters resulted in the LSA similarity score having a Pearson's correlation of -.520 with Rater A. k was found to be optimal at 9.

Algorithm 4.1: PLOTCOMPARE(E, R)

$i \leftarrow 0$
$f \leftarrow \emptyset$
for $e \in E$
  do for $r \in R$
    do { **if** $e = \text{SYN}(r)$
      then $f_i \leftarrow 1$
      else $f_i \leftarrow 0$
    for $n_e \in N_e$
      do { **for** $n_r \in N_r$
        do { **if** $n_e = \text{SYN}(n_r)$
          do { **then** $f_i \leftarrow 1$
            else $f_i \leftarrow 0$
        $i = i + 1$

Figure 1: **Plot Comparison Algorithm**

Classifier	Corpus	Features	% Correct
k-NN	Adventure	LSA	47.5
Naive Bayes	Adventure	PLOT	***55.6***
k-NN	Thief	LSA	41.2
Naive Bayes	Thief	PLOT	*45.4*

Table 2: **Machine-Learning Results**

The results of the plot comparison algorithm were given as features to machine-learners, with results produced using ten-fold cross-validation. A Naive Bayes learner discovers the different statistical distributions of events for each rating. The results for both the "Adventure" and "Thief" stories are displayed in Table 2. "PLOT" means the results of the Plot Comparison Algorithm were used as features for the machine-learner while "LSA" means the similarity scores for Latent Semantic Analysis were used instead. Note that the same machine-learner could not be used to judge the effect of LSA and PLOT since LSA scores are real numbers and PLOT a set of features encoded as binary vectors.

The results do not seem remarkable at first glance. However, recall that the human raters had an average of 56% agreement on story ratings, and in that light the Naive Bayes learner approaches the performance of human raters. Surprisingly, when the LSA score is used as a feature in addition to the results of the plot comparison algorithm for the Naive Bayes learners, there is no further improvement. This shows features given by the event

Class	1	2	3	4
1 (Excellent)	14	22	0	1
2 (Good)	5	36	0	7
3 (Fair)	3	20	0	2
4 (Poor)	0	11	0	39

Table 3: **Naive Bayes Confusion Matrix: "Adventure"**

Class	Precision	Recall
Excellent	.64	.38
Good	.40	.75
Fair	.00	.00
Poor	.80	.78

Table 4: **Naive Bayes Results: "Adventure"**

structure better characterize plot structure than the word distribution. Unlike previous work, the use of both the plot comparison results and LSA did not improve performance for Naive Bayes, so the results of using Naive Bayes with both are not reported (Halpin et al., 2004).

The results for the "Adventure" corpus are in general better than the results for the "Thief" corpus. However, this is due to the "Thief" corpus being smaller and having an infrequent number of "Excellent" and "Poor" stories, as shown in Table 1. In the "Thief" corpus the learner simply collapses most stories into "Good," resulting in very poor performance. Another factor may be that the "Thief" story was more complex than the "Adventure" story, featuring 9 characters over 5 scenes, as opposed to the "Adventure" corpus that featured 4 characters over 2 scenes.

For the "Adventure" corpus, the Naive Bayes classifier produces the best results, as detailed in Table 4 and the confusion matrix in Figure 3. A close inspection of the results shows that in the "Adventure Corpus" the "Poor" and "Good" stories are classified in general fairly well by the Naive Bayes learner, while some of the "Excellent" stories are classified as correctly. A significant number of both "Excellent" and most "Fair" stories are classified as "Good." The "Fair" category, due to its small size in the training corpus, has disappeared. No "Poor" stories are classified as "Excellent," and no "Excellent" stories are classified as "Poor." The increased difficulty in distinguishing "Excellent" stories from "Good" stories is likely due to the use of inference by "Excellent" stories, which our system does not use. An inspection of the rating scale's wording reveals the similarity in wording between the "Fair" and "Good" ratings. This may explain the lack of "Fair" stories in the corpus and therefore the inability of machine-learners to recognize them. As given by a survey of five teachers experienced in using the story rewriting task in schools, this level of performance is not ideal but acceptable to teachers.

Our technique is also shown to be easily portable over different domains where a teacher can annotate around one hundred sample stories using our scale, although performance seems to suffer the more complex a story is. Since the Naive Bayes classifier is fast (able to classify stories in only a few seconds) and the entire algorithm from training to advice generation (as detailed below) is fully automatic once a small training corpus has been produced, this technique can be used in real-life tutoring systems and easily ported to other stories.

5 Automated Advice

The plot analysis agent is not meant to give the students grades for their stories, but instead use the automatic ratings as an intermediate step to produce advice, like other hybrid tutoring systems (Rose et al., 2002). The advice that the agent can generate from the automatic rating classification is limited to coarse-grained *general advice*. However, by inspecting the results of the plot comparison algorithm, our agent is capable of giving detailed fine-grained *specific advice* from the relationships of the events in the story. One tutoring system resembling ours is the WRITE system, but we differ from it by using event structure to represent the information in the system, instead of using rhetorical features (Burstein et al., 2003). In this regards it more closely resembles the physics tutoring system WHY-ATLAS, although we deal with narrative stories of a longer length than physics essays. The WHY-ATLAS physics tutor identifies missing information in the explanations of students using theorem-proving (Rose et al., 2002).

5.1 Advice Generation Algorithm

Different types of stories need different amounts of advice. An "Excellent" story needs less advice than a "Good" story. One advice statement is "general," while the rest are specific. The system

produces a total of seven advice statements for a "Poor" story, and two less statements for each rating level above "Poor."

With the aid of a teacher, a number of "canned" text statements offering general advice were created for each rating class. These include statements such as "It's very good! I only have a few pointers" for a "Good" story and "Let's get help from the teacher" for "Poor" story. The advice generation begins by randomly selecting a statement suitable for the rating of the story. Those students whose stories are rated "Poor" are asked if they would like to re-read the story and ask a teacher for help.

The generation of specific advice uses the results of the plot-comparison algorithm to produce specific advice. A number of advice templates were produced, and the results of the *Advice Generation Algorithm* fill in the needed values of the template. The ϕ most frequent events in "Excellent" stories are called the *Important Event Structure*, which represents the "important" events in the story in temporal order. Empirical experiments led us $\phi = 10$ for the "Adventure" story, but for longer stories like the "Thief" story a larger ϕ would be appropriate. These events correspond to the ones given the highest weights by the Naive Bayes algorithm. For each event in the event structure of a rewritten story, a search for a match in the important event structure is taken. If a predicate name match is found in the important event structure, the search continues to attempt to match the arguments. If the event and the arguments do not match, advice is generated using the structure of the "important" event that it cannot find in the rewritten story.

This advice may use both the predicate name and its arguments, such as "Did the stork fly?" from *fly(stork)*. If an argument is missing, the advice may be about only the argument(s), like "Can you tell me more about the stork?" If the event is out of order, advice is given to the student to correct the order, as in "I think something with the stork happened earlier in the story."

This algorithm is formalized in Figure 2, with all variables being the same as in the Plot Analysis Algorithm, except that W is the Important Event Structure composed of events w with the set of arguments N_w. M is a binary vector used to store the success of a match with index i. The ADV function, given an event, generates one advice statement to be given to the student.

Algorithm 5.1: ADVICEGENERATE(W, R)

for $w \in W$
do
- $M = \emptyset$
- $i = 0$
- **for** $r \in R$
  - **do**
    - **if** $w = r$ **or** $SYN(r)$
      - **then** $m_i = 1$
      - **else** $m_i = 0$
    - $i = i + 1$
    - **for** $n_w \in N_w$
      - **do**
        - **for** $n_r \in N_r$
          - **do**
            - **if** $n_w = \mathrm{SYN}(n_r)$ **or** n_r
              - **then** $m_i \leftarrow 1$
              - **else** $m_i \leftarrow 0$
            - $i = i + 1$
- $ADV(w, M)$

Figure 2: **Advice Generation Algorithm**

An element of randomization was used to generate a diversity of types of answers. An advice generation function (ADV) takes an important event (w) and its binary matching vector (M) and generates an advice statement for w. Per important event this advice generation function is parameterized so that it has a 10% chance of delivering advice based on the entire event, 20% chance of producing advice that dealt with temporal order (these being parameters being found ideal after testing the algorithm), and otherwise produces advice based on the arguments.

5.2 Advice Evaluation

The plot advice algorithm is run using a randomly selected corpus of 20 stories, 5 from each plot rating level using the "Adventure Corpus." This produced matching advice for each story, for a total of 80 advice statements.

5.3 Advice Rating

An advice rating scheme was developed to rate the advice produced in consultation with a teacher.

1. *Excellent*: The advice was suitable for the story, and helped the student gain insight into the story.

2. *Good*: The advice was suitable for the story,

Rating	% Given
Excellent	0
Good	35
Fair	60
Poor	5

Table 5: **Advice Rating Results**

and would help the student.

3. *Fair*: The advice was suitable, but should have been phrased differently.

4. *Poor*: The advice really didn't make sense and would only confuse the student further.

Before testing the system on students, it was decided to have teachers evaluate how well the advice given by the system corresponded to the advice they would give in response to a story. A teacher read each story and the advice. They then rated the advice using the advice rating scheme. Each story was rated for its overall advice quality, and then each advice statement was given comments by the teacher, such that we could derive how each individual piece of advice contributed to the global rating. Some of the general "coarse-grained" advice was "Good! You got all the main parts of the story" for an "Excellent" story, "Let's make it even better!" for a "Good" story, and "Reading the story again with a teacher would be help!" for a "Poor" story. Sometimes the advice generation algorithm was remarkably accurate. In one story the connection between a curse being lifted by the possession of a coin by the character Nils was left out by a student. The advice generation algorithm produced the following useful advice statement: "Tell me more about the curse and Nils." Occasionally an automatically extracted event that is difficult to interpret by a human or simply incorrectly is extracted. This in turn can cause advice that does not make any sense can be produced, such as "Tell me more about a spot?". Qualitative analysis showed that "missing important advice" to be the most significant problem, followed by "nonsensical advice."

5.4 Results

The results are given in Table 5. The majority of the advice was rated overall as "fair." Only one story was given "poor" advice, and a few were given "good" advice. However, most advice rated as "good" was the advice generated by "excellent" stories, which generate less advice than other types of stories. "Poor" stories were given almost entirely "fair" advice, although once "poor" advice was generated. In general, the teacher found "coarse-grained" advice to be very useful, and was very pleased that the agent could detect when the student needed to re-read the story and when a student did not need to write any more. In some cases the specific advice was shown to help provide a "crucial detail" and help "elicit a fact." The advice was often "repetitive" and "badly phrased." The specific advice came under criticism for often not "being directed enough" and for being "too literal" and not "inferential enough." The rater noticed that "The program can not differentiate between an unfinished story...and one that is confused." and that "Some why, where and how questions could be used" in the advice.

6 Conclusion and Future Work

Since the task involved a fine-grained analysis of the rewritten story, the use of events that take plot structure into account made sense regardless of its performance. The use of events as structured features in a machine-learning classifier outperformed a classifier that relied on a unstructured "bag-of-words" as features. The system achieved close to human performance on rating the stories. Since each of the events used as a feature in the machine-learner corresponds to a particular event in the story, the features are easily interpretable by other components in the system and interpretable by humans. This allows these events to be used in a template-driven system to generate advice for students based on the structure of their plot.

Extracting events from text is fraught with error, particularly in the ungrammatical and informal domain used in this experiment. This is often a failure of our system to detect semantic content units through either not including them in chunks or only partially including a single unit in a chunk. Chunking also has difficulty dealing with prepositions, embedded speech, semantic role labels, and complex sentences correctly. Improvement in our ability to retrieve semantics would help both story classification and advice generation.

Advice generation was impaired by the ability to produce directed questions from the events using templates. This is because while our system could detect important events and their or-

der, it could not make explicit their connection through inference. Given the lack of a large-scale open-source accessible "common-sense" knowledge base and the difficulty in extracting inferential statements from raw text, further progress using inference will be difficult. Progress in either making it easier for a teacher to make explicit the important inferences in the text or improved methodology to learn inferential knowledge from the text would allow further progress. Tantalizingly, this ability for a reader to use "inference to grasp points that may not have been explicit in the story" is given as the hallmark of truly understanding a story by teachers.

References

Steven Abney. 1995. Chunks and dependencies: Bringing processing evidence to bear on syntax. In Jennifer Cole, Georgia Green, and Jerry Morgan, editors, *Computational Linguistics and the Foundations of Linguistic Theory*, pages 145–164.

Breck Baldwin. 1997. CogNIAC : A High Precision Pronoun Resolution Engine.

F.C. Bartlett. 1932. *Remembering*. Cambridge University Press, Cambridge.

Johan Bos, Stephen Clark, Mark Steedman, James Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *In Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*. Geneva, Switzerland.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the WRITE Stuff: Automatic Identifi cation of Discourse Structure in Student Essays. *IEEE Intelligent Systems*, pages 32–39.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society For Information Science*, (41):391–407.

Christine Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

A. Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, D. Harter, and N. Person. 2000. Using latent semantic analysis to evaluate the contributions of students in autotutor. *Interactive Learning Environments*, 8:149–169.

Claire Grover, Colin Matheson, Andrei Mikheev, and Marc Moens. 2000. LT TTT - A Flexible Tokenisation Tool. In *Proceedings of the Second Language Resources and Evaluation Conference*.

Harry Halpin, Johanna Moore, and Judy Robertson. 2004. Automatic analysis of plot for story rewriting. In *In Proceedings of Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Maya Hickmann. 2003. *Children's Discourse: person, space and time across language*. Cambridge University Press, Cambridge, UK.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer Academic.

Thomas. Landauer and Susan Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*.

B. Lemaire, S. Mandin, P. Dessus, and G. Denhire. 2005. Computational cognitive models of summarization assessment skills. In *In Proceedings of the 27th Annual Meeting of the Cognitive Science Society*, Stressa, Italy.

Fiona McNeill, Harry Halpin, Ewan Klein, and Alan Bundy. 2006. Merging stories with shallow semantics. In *Proceedings of the Knowledge Representation and Reasoning for Language Processing Workshop at the European Association for Computational Linguistics*, Genoa, Italy.

Erik T. Mueller. 2003. Story understanding through multi-representation model construction. In Graeme Hirst and Sergei Nirenburg, editors, *Text Meaning: Proceedings of the HLT-NAACL 2003 Workshop*, pages 46–53, East Stroudsburg, PA. Association for Computational Linguistics.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *In Proceedings of the Joint Conference of the North American Association for Computational Linguistics and Human Language Technologies*. Boston, USA.

E. Riloff. 1999. Information extraction as a stepping stone toward story understanding. In Ashwin Ram and Kenneth Moorman, editors, *Computational Models of Reading and Understanding*. MIT Press.

Judy Robertson and Beth Cross. 2003. Children's perceptions about writing with their teacher and the StoryStation learning environment. *Narrative and Interactive Learning Environments: Special Issue of International Journal of Continuing Engineering Education and Life-long Learning*.

Judy Robertson and Peter Wiemer-Hastings. 2002. Feedback on children's stories via multiple interface agents. In *International Conference on Intelligent Tutoring Systems*, Biarritz, France.

C. Rose, D. Bhembe, A. Roque, S. Siler, R. Srivastava, and K. VanLehn. 2002. A hybrid language understanding approach for robust selection of tutoring goals. In *International Conference on Intelligent Tutoring Systems*, Biarritz, France.