Story Assembly in the R^{2}aft Dyslexia Fluency Tutor

Arthur Ward
Department of Biomedical Informatics
University of Pittsburgh
Pittsburgh, Pa., 15232
akw13@pitt.edu

Rebecca Crowley
Department of Biomedical Informatics
University of Pittsburgh
Pittsburgh, Pa., 15232
CrowleyRS@upmc.edu

Abstract

To overcome their substantial barriers to fluent reading, students with dyslexia need to be enticed to read more, and to read texts with carefully controlled lexical content. We describe and show examples from a prototype of the new R^{2}aft story assembly engine, which generates an interactive text that has A) variable plot and B) lexical content which is individualized by decoding pattern.

1 Introduction

Dyslexia is a specific disability which prevents students from reading at a level commensurate with their general intelligence. It is also the most common learning disability, affecting as many as 15 (NICHD, 2011) or 20% of the population (Shaywitz, 2003).

We have recently started a new Intelligent Tutoring System project to address dyslexia. The R^{2}aft tutor (**R**epeated **R**eading **A**daptive **F**luency **T**utor) is intended to improve reading fluency among students with dyslexia. An important part of the R^{2}aft tutor will be its story assembly engine **TASA** (**T**ext **A**nd **S**tory **A**ssembler), which will generate the text to be read. In this paper, we will discuss how the special characteristics of dyslexia influenced the design of **TASA**, and describe the prototype system.

Research has shown that phonological processing is the core deficit in dyslexia, but one which can be addressed by intensive training in phonemic awareness and phonics (e.g. (Torgesen et al., 2001)). Because dyslexic readers have difficulty distinguishing individual phonemes within a word, they also have great difficulty learning the relationships between written letter patterns and the sounds they make. These decoding patterns, which are often absorbed intuitively by normal readers, must be explicitly taught to dyslexic readers. For example, the words "tramped" and "padded" both end in "ed," but are pronounced differently. In "tramped" "ed" makes a "t" sound (the "ed_t" pattern), while in "padded" it makes an "ed" sound (the "ed_ed" pattern). [1] A dyslexic reader must receive explicit training on hundreds of decoding patterns such as these. In addition, to improve fluency a dyslexic reader must practice these patterns extensively, in large amounts of connected text.

Unfortunately, this practice is difficult to obtain. "Decodable texts" which use a constrained vocabulary are available (e.g. (Bloomfield et al., 1998)), however, professional dyslexia tutors report that these booklets often do not meet the individual vocabulary needs of their students. In addition, students with dyslexia typically *hate* to read (Shaywitz, 2003), and do little to acquire the necessary practice in connected text.

This analysis suggests that a successful fluency tutor should address two sets of issues. It should address vocabulary issues to ensure that the student gets practice on appropriate decoding patterns. It also should address motivational issues, to entice students to read more text. As described below, **TASA** addresses the vocabulary issue by using templates whose slots allow for lexical individualization. It aims to improve motivation by generating a variable plot line, and allowing students to make

[1] These examples are taken from (Bloomfield et al., 1998).

Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications, pages 130–135,
Portland, Oregon, 24 June 2011.

plot choices that affect the unfolding story. There are several reasons to expect that allowing dyslexic students to interactively shape plot events will improve their motivation. For example the popular "Choose Your Own Adventure" books (e.g. (Montgomery, 1982)), allow their readers to choose paths in a branching narrative. Also, "Interactive Fiction" type text adventures (Montfort, 2003) which allow the reader to control the protagonist, enjoy continuing popularity.[2] Furthermore, a study with the REAP ESL vocabulary tutor suggests that presenting more interesting stories can improve learning (Heilman et al., 2010).

Authoring a branching narrative entirely by hand, however, is an unattractive option. Even a small story could require authoring hundreds of plot branches. Reducing this burden would allow authoring the large volume of text needed by our readers.

The dyslexia tutoring domain therefore suggests three design goals for our story generation engine.

1. **Lexical Individualization:** It should allow us to fine tune lexical content to feature the decoding patterns required by each student.
2. **Interactivity:** It should allow student plot choices to influence the story being read.
3. **Tractable Authoring:** It should help reduce the burden of authoring multiple story branches.

2 Story Representation

As described above, the dyslexia domain requires detailed control over lexical content in our reading, and our approach to motivational issues involves interactive text. Both of these considerations argue against the use of pre-existing texts as are used, for example, in the Listen (Mostow, 2011) or REAP (Brown and Eskenazi, 2004) systems. Instead, we investigate generating our own stories.

The literature on story analysis and generation can be usefully divided into approaches which model the structure of the story itself, versus approaches which model some of the processes involved in story creation. The latter often simulate the author (e.g. (Dehn, 1981), the reader (Bailey, 1999), or the story world (Meehan, 1976). They also typically require large amounts of real-world knowledge to generate even simple stories. Given our need to produce a large amount of practice text, this approach seems untenable.

Instead, we take the first mentioned approach of modeling story structure. To do this, we will require a formalism which will allow us to represent, manipulate and re-combine pre-written stories. Early work in story grammars used elegant hierarchical tree structures to *analyze* plot structure. (see (Graesser et al., 1991) for an overview of this work). In general, these structures seem underspecified for *generating* stories.

We turn instead to the causal network theory of Trabasso and van den Broek (1985). This formalism does not enforce strict hierarchies, but represents text as a sequence of nodes. The nodes represent categories such as "Setting," "Goal," "Event," "Reaction" or "Outcome," and are connected in sequence by temporal and causal links.

This formalism provides guidelines about legal node sequences. It also enforces several constraints on the type of text we can represent. For example, the text base must be generated in strict temporal and causal order, we cannot represent flashbacks or other types of transformed narrative. Also, each node generates text, we cannot represent events which do not appear in the textbase.

3 The Story

Our prototype story is a rehash of several standard themes common in young reader fiction. A young protagonist moves to a new house with a parent (the genre seems to require that one parent be missing). This protagonist is shown to be weak and fearful in various ways. The protagonist discovers some source of inspiration which leads him/her to attempt some endeavor. After many setbacks the protagonist becomes accomplished at this endeavor, then at the climax uses his/her new strength/skill to save the parent from certain doom.

Our prototype story was developed to instantiate these themes each in several ways. For example, the initial source of inspiration comes in two options. The first source will be found in a springhouse at the rear of the protagonist's new home. The alternate source is found in a locked room of the main house. There are also several options for the resulting en-

[2] However, see (Glassner, 2004, pg 239) for a discussion of the difficulties of branching narrative.

deavor and several options for the final climax. Pursuant to our goal of presenting an *interactive* text, most of these plot variations will be determined at runtime by reader choice.

For example, toward the beginning of the story, the protagonist is found in his/her new bedroom, with a goal to explore the unfamiliar house. Here the reader chooses what to investigate, with the next plot fork being determined by whether the "springhouse" or the "locked door" is chosen.

This story is written in text form, then deconstructed into a causal network in the following way. An analyst examines the initial story text, and divides it into chunks. Following the Trabasso framework described in Section 2, each chunk is required to be temporally and causally subsequent to the previous chunk, and to depict elements such as a "setting," "event," "goal," "attempt" "reaction" or "outcome." The story chunk described above, for example, is labeled as a goal node. The subsequent chunk in which the protagonist begins to explore, is labeled as an "attempt." After this analysis, the resulting chunks are instantiated as production system rules, as described below.

4 The TASA Prototype

Our prototype TASA system is instantiated as a set of facts and rules in the Clips expert system shell (Giarratano and Riley, 1994). Expert systems typically consist of a set of *if-then rules*, plus a set of *facts* asserted in memory. Rules whose *if* portions are satisfied by facts are activated and placed on an *agenda*. A rule on the agenda is then selected and fired, according to some salience scheme. Rules typically assert new, or modify old facts in memory. These facts then cause more rules to activate and fire, and the cycle continues until the agenda is empty. In our system, we write rules which append text to the accumulating story when the story world is in a particular state.

The TASA system includes three types of facts: user-model, story-world, and lexicon facts.

User-model facts include details about the student's age and gender, as well as about targeted decoding patterns for that individual student.

Figure 1 shows an abbreviation of a student fact. This fact records information about the current student user such as age and gender. It also records the set of decoding patterns that should be selected in the text. The "ed_t" pattern is shown.

```
(student
    (decodePat ed_t)
    (age 9)
    (gender m))
```

Figure 1: Abbreviated student fact, requesting "ed_t" pattern

Story-world facts include the text so far, as well as the relevant story state. The story state is much less detailed than is required for the story generation systems described in Section 2, and simply includes information about the location, goals and mood of characters, and the locations and status of certain objects, as seems necessary to prevent rules from appending text in inappropriate places. It prevents, for example, text about unlocking a door from being appended when the door is open. Facts belonging to the same world state are co-indexed (with the "worldHist" variable shown in Figure 1), so that when a rule modifies the world state, the entire set of world facts can be re-asserted into memory with an updated index. This allows several different plot branches to be developed in memory simultaneously, without context-breaching intrusions from each other.

```
(character
    (charID clif_1)
    (worldHist 0)
    (role protag)
    (firstName Clif)
    (gender m)
    (goal explore_springHouse)
    (location bedroom)
    (subjPronoun he)
    (objPronoun him)
    (posPronoun his)
    (age 9))
```

Figure 2: Abbreviated story-world fact for protagonist

Figure 2 abbreviates a "protagonist" story-world fact. Among other things, this fact contains the protagonist's current location and goal, as well as appropriate forms for pronominal reference. Note also that the protagonist's age and gender have been set to match those of the current student user.

The **Lexicon facts** are a large set of words known

to the system. Each word is associated with both a synonym and a decoding pattern. This allows the system to locate all appropriate substitutions for a target word which also exhibit a targeted decoding pattern.

```
(decodeSet
    (decodePat ed_ed)
    (word padded)
    (syn walked))
(decodeSet
    (decodePat ed_t)
    (word tramped)
    (syn walked))
```

Figure 3: Example Lexicon Facts

Figure 3 shows several example lexicon facts. They allow the system, for example, to locate words which can substitute for "walked" and which display the "ed_ed" decoding pattern. Note that organizing our lexicon by substitutable synonyms allows the prototype to dispense with representing things like tense and number. Other senses of "walked," if needed, would be listed with an index, ie. "walked_2."

In the final system, we expect to implement the 70 to 80 decoding patterns commonly featured in Orton-Gillingham (Orton-Gillingham, 2011) based instructional materials. Based on discussions with professional dyslexia tutors, we hope to provide at least five examples of each pattern, requiring a lexicon of above 400 words. In addition, we hope to show each example word in several sentence contexts, which brings the number of expected sentence templates well into the thousands.

As mentioned above, each node from the story analysis is instantiated as one or more rules in this expert system. If a rule matches a story-world state and fires, it appends text to the story so far. Each rule also changes the story-world in some feature which is modelled by the system and matched in the rule's *if* part. For example a rule should leave the protagonist in a different place or in a different mood than in the previous chunk. This is a practical requirement to prevent the same rule from repeatedly firing when the story-world facts are re-asserted.

The *then* portions of these rules contain templates which are used to generate text. Each template includes slots which are to be filled by appropriate words from the lexicon. Because one template typically does not exhaust all the ways to express the rule's intended message, the analyst typically writes several forms of the rule, which increases the range of potential word use.

Given this structure of rules and facts, the production rule paradigm is appealing for its ability to meet all three of our design goals: plot variety, lexical individualization, and tractable authoring. By modularizing chunks of text and associating them with appropriate story-world conditions (in their *if* parts), we can make a system able to generate plot forks by matching two potential child nodes to the previous story node. We can achieve lexical individualization by writing rules which match not only story world facts, but also student model facts about targeted decoding patterns. Authoring burden is reduced by the ability of existing rules to add text in new situations. We give examples of each of these features below.

As an example of how this works in our prototype system, consider the plot node described above. The protagonist is in the bedroom. If the reader chooses to investigate the springhouse, a rule like the following is activated. [3]

```
(defrule walkAcrossYard_1
    (Code which binds state variables omitted here)
    ?prot ← (character (charID ?proID)
                (worldHist ?rh)
                (location ?proLoc&bedroom)
                (goal explore_springHouse)
                (firstName ?proFn))
          (student (decodePat ?dp))
          (decodeSet (decodePat ?dp)(syn walked)(word ?wlkd))
⇒
(Code which duplicates state variables omitted here)
     (text (str-cat ?txt ?proFn " " ?wlkd " across the back yard to the springhouse. )))
```

Figure 4: Rule describing walk to springhouse

Figure 4 abbreviates a rule which fires if A) the protagonist is in the bedroom with goal to explore the springhouse, and B) the current student needs a decoding pattern ?dp which is available in a synonym of "walked." If these conditions are met, then

[3]For clarity, example rules are extensively pruned from the Clips rule syntax.

(below the ⇒) the rule fills a sentence template with the name of the protagonist and the appropriate synonym of walked.

For example, if the protagonist's name is set to "Clif," (as in Figure 2) and the decoding rule "ed_t" is targeted (as in Figure 1), this rule will produce a sentence for each matching synonym of "walked," (one of which is shown in Figure 3) including:

> Clif tramped across the back yard to the springhouse.

If the targeted pattern had instead been "ed_ed," this rule would produce sentences like "Clif padded across the back yard to the springhouse."

When the rule fires, the story world is changed to place Clif at the springhouse door, which causes additional rules to be activated. Still assuming the "ed_ed" decoding rule is active, one subsequent rule appends a sentence as follows:

> Clif hunted across the back yard to the springhouse. He pounded on the door, and listened for an answer.

Alternatively, if the source of inspiration in the story is set to be in the locked room, **TASA** produces a different variety of sentences including:

> Clif padded across the room toward the locked door. He pounded on the door, and listened for an answer.

Note from Figure 3 that "padded" is in the lexicon as another synonym for "walked" that follows the "ed_ed" decoding pattern. Also note that in this example the second sentence was produced by the same rule that provided the second sentence in the previous example, which had been written for a different branch of the plot. Together, these examples show how **TASA** can provide both plot variation and lexical individualization. They also demonstrate the feature of text reuse, which we expect will become more prevalent as the rule base grows larger.

5 Future Work

In our ongoing work we are re-implementing the prototype system in the Drools expert system shell (Bali, 2009). Drools provides for the inclusion of Java code in instantiated story-world facts, which will allow us to offload the substantial portion of our prototype rules devoted to updating and maintaining the story state. In addition we are greatly expanding our rule-base as we instantiate more of the prototype story. In the course of this work we will also evaluate moving to a more expressive story formalism, such as Graesser's Conceptual Graph Structures (Graesser et al., 1991) which can represent additional relationships between nodes.

In addition, we will evaluate improved ways to select the best text from the many options output by the system. Rather than simply comparing the number of targeted decoding patterns (as we do now) we will experiment with other evaluation metrics such as cohesion (Graesser et al., 2004), or methods which have been useful in essay evaluation (e.g. : (Higgins et al., 2004)).

After sufficient story development, we intend to evaluate the effect of interactive text on students' motivation to read. This evaluation will collect motivational survey results and "voluntary" reading times, and compare them between students using interactive and non-interactive versions of the system.

Acknowledgments

This work was supported by the National Library of Medicine Training Grant 5 T15 LM007059. We also thank Carol Utay for several productive conversations about dyslexia tutoring.

References

Paul Bailey. 1999. Searching for Storiness: Story-Generation from a Reader's Perspective. *Proceedings of the AAAI Fall 99 Symposium on Narrative Intelligence*.

Michal Bali. 2009. *Drools JBoss Rules 5.0 Developer's Guide*. Packt Publishing Ltd., Birmingham, B27 6PA, UK.

Leonard Bloomfield, Clarence Barnhart, Robert Barnhart, and Cynthia Barnhart. 1998. *Let's Read 7 Revised Edition*. Educators Publishing Service, Inc., Cambridge, MA, USA.

Jonathan Brown and Maxine Eskenazi. 2004. Retrieval of authentic documents for reader-specific lexical practice. In *In Proceedings of InSTIL/ICALL Symposium*.

Natalie Dehn. 1981. Story generation after tale-spin. In *In IJCAI-81*, pages 16–18.

Joseph Giarratano and Gary Riley. 1994. *Expert Systems: Principles and Programming*. PWS Publishing Co., Boston, MA, USA.

Andrew Glassner. 2004. *Interactive Storytelling: Techniques for 21st Century Fiction*. A.K. Peters, Nantick, MA.

Arthur Graesser, Jonathan Golding, and Debra Long. 1991. Narrative representation and comprehension. In Rebecca Barr, Michael Kamil, Peter Mosenthal, and P. David Pearson, editors, *Handbook of Reading Research, vol. 2*, pages 171 –205. Longman Publishing Group, White Plains, NY.

Arthur Graesser, Danielle McNamara, Max Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, and Computers*, 36:193–202.

Michael Heilman, Kevyn Collins-Thompson, Jamie Callan, Maxine Eskenazi, Alan Juffs, and Lois Wilson. 2010. Personalization of reading passages improves vocabulary acquisition. *International Journal of Artificial Intelligence in Education*, 20:73–98, January.

D. Higgins, J. Burstein, D. Marcu, and C. Gentile. 2004. Evaluating multiple aspects of coherence in student essays. *Proceedings of the Annual Meeting of HLT/NAACL*, pages 185 – 192.

James Meehan. 1976. *The Metanovel: Writing Stories by Computer*. Doctor of philosophy, Yale University, New Haven, Conn.

Nick Montfort. 2003. *Twisty Little Passages: an approach to interactive fiction*. MIT Press, Cambridge, Massachusetts.

R. A. Montgomery. 1982. *The Abominable Snowman*. Chooseco, LLC., Waitsfield, Vermont.

Jack Mostow. 2011. Project listen: A reading tutor that listens. http://www.cs.cmu.edu/ listen/.

NICHD. 2011. What are learning disabilities? http://www.nichd.nih.gov/health/topics/learning_disabilities.cfm.

Orton-Gillingham. 2011. Institute for multi-sensory education. http://www.orton-gillingham.com/.

Sally E. Shaywitz. 2003. *Overcoming Dyslexia: a new and complete science-based program for reading problems at any level.* Vintage Books, New York.

Joseph Torgesen, Ann Alexander, Richard Wagner, Carol Rashotte, Kytja Voeller, and Tim Conway. 2001. Intensive remedial instruction for children with severe reading disabilities: immediate and long-term outcomes from two instructional approaches. *Journal of Learning Disabilities*, 34:33–58,78.

Tom Trabasso and Paul van den Broek. 1985. Causal thinking and the representation of narrative events. *Journal of Memory and Language*, 24:612 – 630.