Random Word Retrieval for Automatic Story Generation

Richard S. Colon, Sr., Prabir K. Patra, and Khaled M. Elleithy

Abstract— Over the past forty years, significant research has been done on story/narrative generation in which the computer is the author. Many existing systems generate stories by filling in a template or copying an analogous story (and changing the time, place, etc.) or by prompting the user to provide a start to the story. Very few systems generate variable stories without these techniques. While it is impossible to quantify a human writer’s inspiration, we can consider a common exercise that authors perform; namely ‘writing prompts’. A writing prompt is just a topic or idea around which to start writing. The prompt can simply be a few words, which becomes the basis for a story. In this paper we present story generation from the perspective of how human authors create stories via writing prompts. The system will select a few random words as a prompt, which will form the basic parameters for generating a story. But unlike a human writer, a computer cannot intuitively know the context of a chosen word. Therefore, the Internet (and existing ‘Concept Knowledge’ systems) will be used to find the context for the selected words, thus guiding the story generation process.

Index Terms—Computational Creativity, Story/Narrative Generation, Knowledge Based Systems, Expert Systems

I. INTRODUCTION

One of the most intriguing topics in computer science research is creativity. To be more specific, can we program machines to be creative? In an attempt to answer this question, there is a great deal of ongoing work in the area of Computational Creativity, with publications, dedicated conferences and workshops [1] [2] [3]. A popular definition of Computational Creativity is:

“The study and support, through computational means and methods, of behavior exhibited by natural and artificial systems, which would be deemed creative if exhibited by humans.” [4]

If we consider the activity of creating literature, can a computational system write a story such that a reader would not know the story was computer generated? Can the stories be varied and random enough to be interesting and non-repeatable? Will these generated stories evoke emotion in the human reader?

The goal of computational creativity is to model, simulate or replicate creativity using a computer, to achieve one of several ends:

• to construct a program or computer capable of human-level creativity
• to better understand human creativity and to formulate an algorithmic perspective on creative behavior in humans
• to design programs that can enhance human creativity without necessarily being creative themselves

During the many years of research, a number of successful story/narrative generation systems have been created. Most of the existing systems have one or more of the following characteristics:

• Self-contained corpus of knowledge not utilizing the Internet
• Designs based upon using analogy or user input to drive the narrative
• Only generates a specific genre such as folk tales, poetry, suspense, etc.
• Functions primarily as a problem-solver

The research being presented here approaches fictional short story generation from the perspective of how human authors create stories via random word prompts, but the goal is to focus solely on the production of creative results. An important consideration for this work is story randomness without user intervention. That is the reason for selecting the 'random words' technique for writing a story. However, if the system design is closed and self-contained, then eventually stories might be regenerated or may seem derivative. Therefore, to gain a wider knowledge base, it is only logical to access the Internet to aid in story generation.

The ‘knowledge’ required to generate even a simple story is not trivial. Consider basic concepts/facts such as dogs bark, birds fly, and fire burns. We take for granted our human knowledge about the world and causal relations (doing X, results in Y). But a machine has no understanding of word concepts within any specific context. Therefore, many early story/narrative generation systems had difficulty handling
even the simplest forms of cause and effect situations. However, we have a significant advantage today, with the ease and availability of the Internet along with existing ‘Concept Knowledge’ systems.

II. IMPORTANT GROUND BREAKING SYSTEMS

A. TAIL-SPIN

One of the most widely referenced research works, is James Meehan’s TALE-SPIN, which was his Ph.D. dissertation at Yale University in 1976 [5]. TALE-SPIN creates a world model consisting of humans and talking animals. Their motivations are essentially physical needs (hunger, thirst, rest, etc.). Meehan viewed the program as a problem-solver which simulated rational behavior by characters in a setting.

While this system was revolutionary for its time, the user had to select the characters and the setting. In addition, the characters had to be assigned a goal. TALE-SPIN determined the relationships between the characters and their personality traits. As the characters attempted to achieve their goals they would interact and take action. These events would have consequences which then led to other events, and thus the characters would take other actions. These events, consequences, and actions, become a story. Fig. 1 shows a sample of the original output from TALE-SPIN.

B. MINSTREL

Now let’s move forward almost 20 years to look at MINSTREL. Scott Turner developed MINSTREL which is a large and complex program that generates short ‘themed based’ stories about King Arthur and the Knights of the Round Table [6][7]. Minstrel generates a story by following a problem solving process. It employs a case-based reasoning approach, which uses past problem-solving situations (cases) to solve current problems.

The main issue with case-based storytelling is how to avoid repeating a story since we are using previous similar story knowledge. To overcome this problem, MINSTREL implements a search and adaptation process called TRAMs (Transform-Recall-Adapt-Methods) to transform and adapt an existing recalled story element into a new story element.

Referring to Fig. 2, we see another example of automatic story generation that is not exactly great literature, but it is still an adequate story. Turner realized the limited quality of the stories generated, but he wanted more attention to be paid to how the system creates stories not just on what is created. The problem solving ability of MINSTREL was its main focus, not necessarily quality story generation.

C. BRUTUS

Selmer Bringsjord and David A. Ferrucci developed BRUTUS as a program that tells stories which are based on the theme of betrayal [8]. Unlike Meehan or Turner, Bringsjord and Ferrucci did not focus on studying the "how" of story generation; they focused more on the quality of the story that was being generated. As you will see, the language and coherence of BRUTUS goes beyond what other systems have accomplished. A sample generated story is shown in Fig. 3.
Dave Striver loved the university. He loved its ivy-covered clocktowers, its ancient and sturdy brick, and its sun-splashed verdant greens and eager youth. He also loved the fact that the university is free of the stark unforgiving trials of the business world —only this isn’t a fact: academia has its own tests, and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, to become a doctor, one must pass an oral examination on one’s dissertation.

Dave wanted desperately to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless inscription which, together, would certify that he had passed his defense. One the signatures had to come from Professor Hart. Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Striver that it was absolutely first-rate, and that he would gladly sign it at the defense. They even shook hands in Hart’s book-lined office. Dave noticed that Hart’s eyes were bright and trustful, and his bearing paternal.

At the defense, Dave thought that he eloquently summarized Chapter 3 of his dissertation. There were two questions, one from Professor Rodman and one from Dr. Teer; Dave answered both, apparently to everyone’s satisfaction. There were no further objections. Professor Rodman signed. He slid the tome to Teer; she too signed, and then slid it in front of Hart. Hart didn’t move. “Ed?” Rodman said. Hart still sat motionless. Dave felt slightly dizzy. “Edward, are you going to sign?” Later, Hart sat alone in his office, in his big leather chair, underneath his framed PhD diploma.

While seemingly impressive, BRUTUS only generates stories that follow the theme of betrayal. First the theme is instantiated, then the plot is developed (a process where the characters attempt to achieve their goals), and finally generation of the story grammar (descriptions of how to create phrases and sentences).

Some consider BRUTUS as a system that does not represent any creative process. And that is all right with the authors; who themselves argue that simulation of human creativity is not possible. They are satisfied just to create the illusion of creativity.

III. SYSTEM DESIGN

While different Artificial Intelligence techniques have been applied to story generation, the design being presented here is based on the ‘Expert Systems’ approach. “By definition, an Expert System is a computer program that simulates the thought process of a human expert to solve complex decision problems in a specific domain [9].”

Expert Systems are used to solve problems by reasoning about knowledge like an expert, and not by following the fixed procedures of conventional computer programming.

The components of a typical Expert System are as follows:

**Knowledge Base**
- A rule base representation of the expertise, including known facts, heuristic knowledge, ‘rule of thumb’, ‘best guess’.

**Working Memory**
- ‘Facts’ base, containing problem specific current data.

**Inference Engine**
- The rule engine, derives the reasoning (makes logical inferences) from the Knowledge Base and the current data in Working Memory.

**User I/F**
- Sometimes (not always) considered a component of the system.

The fundamental concept is to have problem-solving done by applying specific knowledge, instead of specific techniques. The problem data is stored as ‘facts’ in memory and the system can ‘reason’ using IF <condition> THEN <conclusion> rules. A rule can only execute (known as ‘firing’) when the <condition> specified is true.

The firing of the rule performs the <conclusion> which may be an action, decision, or a new fact. This ‘reasoning’ can be deductive (known as ‘Forward Chaining’) or inductive (‘Backward Chaining’).

These Knowledge Based Systems (and specifically Rule Based Expert Systems), differ from traditional software system implementations. Rather than technical, the differences are more philosophical. Traditional systems focus on algorithms, combining facts and how to use them for a specific purpose. In a Knowledge Based System, the facts are clearly separated from the operations. The representation of knowledge is declarative instead of procedural. While the form and internal structure of Knowledge Based Systems seems relatively simple, it is a complex and tedious task to design and implement a working system [10].

Development of our ‘Story Generation System’ also involves the use of readily available online linguistic support systems:

- **ConceptNet** - a ‘commonsense knowledge’ database which is used to perform textual-reasoning in order to make sense of the everyday world [12]
- **Natural Language Toolkit (NLTK)** - an open source Python library for Natural Language Processing [13]. Python is a high-level programming language (developed in the late 1980s), that provides excellent text processing tools; making it ideal for natural language applications.

Utilizing these existing tools, a functional block diagram of the proposed system is shown in Fig 4. The system will:
- Select a few random words
- Look up the concepts of the selected words
- Search among the different concepts found for each word, and try to correlate what may be common
- Using this information as a guide, generate a basic plot and characters
- Perform detailed story Generation/Evaluation (an iterative process)
IV. RESULTS

To date, the searching components have been implemented and tested to gather sample data:

**Search WordNet**
- Select three random words

**Search ConceptNet** (input: the selected words)
- Based on the selected words retrieve their context

Written in the Python programming language, the code reads three random words from the WordNet corpus. Using WordNet gives the advantage of selecting words by POS (part of speech) and frequency of use.

For the initial tests, we selected three random words comprised of two nouns and one verb. Since WordNet provides multi-word and hyphenated phrases, these have been filtered out for our tests. The following are three sample outputs from executing the word selection code:

(NOTE: WordNet returns a 'synset' that contains: the word, "part of speech", and the sense count.)

```python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
>>> ================ RESTART ================
>>> 
Random Word Selection -
Nouns:
Synset('sleeper.n.09')
Synset('handwriting.n.02')
Verb:
Synset('slump.v.03')
>>> ================ RESTART ================
>>> 
Random Word Selection -
Nouns:
Synset('lighting.n.03')
Synset('intonation.n.04')
Verb:
Synset('clarify.v.02')
>>> ================ RESTART ================
>>> 
Random Word Selection -
Nouns:
Synset('hibernation.n.03')
Synset('moderation.n.04')
Verb:
Synset('jog.v.06')
```

Once we have our three random words, we need to search for the possible contexts for each word. With some additional coding, we parse the WordNet synset to get the selected word and search ConceptNet. The following is a sample output from the program that selects three random words and retrieves context information for each word from ConceptNet:

```python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
>>> ================ RESTART ================
>>> 
Random Word Selection -
Nouns: Synset('hibernation.n.03')
Synset('moderation.n.04')
Verb: Synset('jog.v.06')
```

When Plot Generation is complete, it will forward the plot for evaluation and final story generation.

When the goals of story evaluation are satisfied, the system outputs a completed short story.
Further development, will be the software component that has the difficult task of performing “Concept Correlation”. From the three randomly selected words, we now have a list of concepts. Which of the concepts should be selected? Our selection criteria should be based upon some cohesion between the concepts and potential for story generation. From a human perspective, we can see the correlation between:

**Search word - orchestration**
- Concept: orchestration
- Concept: orchestrate

**Search word - inscription**
- Concept: inscription
- Concept: inscribe

**Search word - erase**
- Concept: erase

These concept selections could generate an interesting story about an evil mastermind who plans to kill his mortal enemy and has already inscribed the tombstone. Our ultimate goal is for the system to derive a similar (or at least coherent) storyline.

V. CONCLUSION

Many of the existing ‘Story Generation Systems’ are problem solvers, and the stories generated are just a path through the problem-solving process. Their designs share some common approaches/limitations such as: computer interaction, genre specific, story copy/adaptation, case-based reasoning, and autonomous agents. The research presented here will place the emphasis on the story itself, and will not necessarily be concerned with tracing a path from start to finish during the generation process. Creation of stories that have the quality of a human author is the main focus; we are not trying to determine how humans create.

One of the most important goals of the proposed system is story randomness. The generation of stories that are not
adapted from existing stories, or that fit into a pre-existing schema, or that are started/driven by a user.

The results presented in this paper are clearly just the beginning. However, this work forms the foundation for development of additional and more complex components. Based on the randomly selected words and contexts, the system needs to generate a setting and characters. Then develop a plot, and start the text generation of a story. No doubt that there is still a long way to go.

Can we begin with random words and develop a cohesive narrative that would be considered a “good” short story? That is the proposed research; that is the challenge being undertaken.

REFERENCES


