A Comparison of Retrieval Models for Open Domain Story Generation

Reid Swanson and Andrew S. Gordon

Institute for Creative Technologies
University of Southern California
13274 Fiji Way, Marina del Rey, CA 90292 USA
swansonr@ict.usc.edu, gordon@ict.usc.edu

Abstract

In this paper we describe the architecture of an interactive story generation system where a human and computer each take turns writing sentences of an emerging narrative. Each turn begins with the user adding a sentence to the story, where the computer responds with a sentence of its own that continues what has been written so far. Rather than generating the next sentence from scratch, the computer selects the next sentence from a corpus of tens of millions of narrative sentences extracted from Internet weblogs. We compare five different retrieval methods for selecting the most appropriate sentence, and present the results of a user study to determine which of these models produces stories with the highest coherence and overall value.

Introduction

The automated generation of fictional stories has proven to be an extremely challenging problem in Artificial Intelligence. Recent work in this area has produced impressive results in narrow domains (Riedl and Leon 2008; Cheong and Young 2006; Riedl and Young 2004), but has been limited by the availability of knowledge resources that support open-domain reasoning about events, as well as the difficulties in generating fluid natural language from these formalisms. Previously, we introduced a new approach to story generation that attempts to address these limitations by casting the problem as a massive-scale collaborative writing project (Swanson and Gordon 2008). In this approach, a user and a computer collaboratively author new stories in an interactive fashion, where each repeatedly takes turns contributing new sentences to an unfolding narrative. Whereas sentences contributed by the human user are limited only by their creativity, the sentences provided by the computer are selected from tens of millions of narrative sentences automatically extracted from Internet weblogs. Despite the simplicity of this approach and the lack of restrictions on the narrative domain, the stories that can be generated are often both coherent and entertaining, as in the following example:

I came home from work and my neighbor had cut down all of the trees in my yard. The magnificent huge fir tree in the backyard hit the neighbor's roof. I guess that made him mad enough to cut down all my trees when I was gone at work. I was relieved to find that he had only just left and was still in sight. I picked up a shovel and crept up behind him. He reached for the door handle and began to turn it slowly before wrenching the door open. I used all my strength and wacked him on the head with the shovel. It was covered in blood and maggots. Someone had used the same shovel earlier to bury their dog who was hit by a car. The wife says I have a bad attitude. Let's see how she feels when some cuts down all the trees in her yard.

The original system (*Say Anything*) described by Swanson and Gordon (2008) employed an extraordinarily simple strategy for selecting an appropriate sentence to contribute during the computer's turn. Although these early results were promising, this work highlighted a number of problems that limited the coherence and quality of the stories. To tackle these problems, we developed a new version of the *Say Anything* system that would allow us to investigate issues of coherence and quality by evaluating the performance of five competing retrieval models.

This paper describes our revision of the *Say Anything* story generation system and presents the results of our evaluation of competing retrieval models. We begin by describing the new system architecture and user interface, followed by a description of the five competing retrieval models of our evaluation. We then present our evaluation of these models, first through a qualitative analysis of several example stories produced by different models, and second through a quantitative analysis of user judgments and authoring behavior.

Architecture

The *Say Anything* architecture follows a simple approach, illustrated in figure 1. It is a cyclical process involving a human and computer contributing alternating sentences to a narrative story. Each turn begins with a human user writing a sentence and is completed by the computer returning a sentence in response. The computer performs three major operations in an attempt to generate a sentence that is cohesive with what has been written so far. First, it

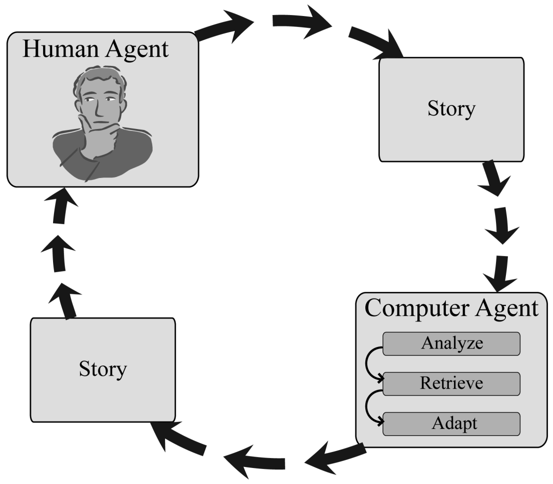


***Figure 1**: Illustration of the story generation architecture.*

analyzes the story, including the user's current input. This analysis is used to retrieve a sentence from a large story database. A story in the database is selected based on a measure of similarity to the user's current input. The retrieved story is treated as an approximation of the current state in the user's story. The next sentence of this story is then chosen as the continuation of the user's developing narrative. To better integrate this sentence into the user's story, a post-processing adaptation step is performed. In this section we discuss the user interface to our system and the main components of the generation process in more detail.

User Interface

To facilitate the authoring of stories, we designed a web-based interface to the *Say Anything* system. The main screen of this user interface provides a workspace for collaboratively authoring a new story, where the user's burgeoning story is presented near the top of the page. A text box for typing the next sentence of the story, along with controls allowing the user to continue or finish the story, is positioned below this text box. After entering a new sentence, it is appended to the bottom of the story, followed by a sentence selected by the computer to move the story forward. At any time during this process, the user may click on a computer-generated sentence, revealing a pop-up menu that presents nine other alternate sentences that the user can choose from instead. We believed this functionality would give users more opportunity to continue their story in case a highly irrelevant or objectionable sentence was initially returned. It also allowed us to conduct a more fine-grained analysis of coherence without having to bother the user with judgments after every sentence. Finally, once the user chooses to end their story, they are asked to give it a title and rate it on two criteria, which are described in the evaluation section of this paper.

Although story generation is the primary focus of our system, we also included several other elements in our interface that we believed would improve our research and make the system more enjoyable for the user. Not only did we want to collect ratings from the author of a particular story, but we also wanted feedback from users of our system who did not write the story. To accomplish this we created a separate page where users could read and rate stories written by other users. Finally, to help encourage our users to participate with the system we included two additional pages; one in which they could read the top ranked stories and another in which they could view all of their previously written compositions.

Story Database

There are two general options in the design of a story database. On one hand we could randomly sample a set of real world stories and construct a relatively small but highly structured formal model from an analysis of these stories. On the other hand we could maintain a relatively large unstructured database by leaving the stories relatively intact. A small highly structured library enables a more fine-grained control over how the system interprets and responds to the input. For example, formal representation of events and predicates in the domain would allow for deep interpretation and well defined rules for progressing to allowable states in the narrative development. For example, Mueller (2006) proposes such a knowledge-base in the domain of restaurant activities.

Despite the potential benefits of a small set of structured representations, there are a few drawbacks that make this approach insufficient for our purposes. Although it is usually good to have precise knowledge and control over state transitions, in the case of narrative generation, too much control is potentially detrimental to the quality of the story and the experience of writing it. Often it is precisely the unexpected outcomes that engage us in a story that would otherwise be the same boring sequence of events that have been rehashed a thousand times before. While not unworkable in a formal representation, it is also not obvious how to solve the problem effectively or efficiently. The other, more serious issue is a matter of scale. Although impressive in its breadth and depth, Mueller's formalization of the restaurant domain supports only one of an intractable number of activities one could tell a story about. Even if we limit ourselves to only the most frequent activities, the number of existing formal theories is unsuitably small. Unfortunately, even with significant effort, it is extremely unlikely that sufficient domain theories could be authored to cover the possible range of activity contexts needed to enable open-domain storytelling.

Instead, we opted to use a large collection of unstructured data. While it is difficult and time consuming to write formal theories of commonsense and mundane activities, there is virtually no limit to the amount of data written about these topics on the web in natural language. Our database is derived from a large collection of stories harvested from Internet weblogs by Gordon, Cao and Swanson (2007). This corpus consists of 3.7 million segments of weblog text (66.5 million sentences), classified as story-like text using statistical text classification techniques.

Using this story collection for narrative generation might seem, at first, problematic for two related reasons. Because our database is unstructured, it is probably difficult for the computer to have any deep understanding of the stories without applying advanced natural language processing techniques. Without this understanding it seems impossible to ensure that similar cases can be found, or to guarantee that what happens next in these similar cases will be coherent in the context of the user's developing narrative. Compounding the problem is that, due to the scale of our corpus, sophisticated NLP techniques for extracting rich structure would require an intractable amount of computational resources. While it would be important to have enough structure for some level of deep understanding, it is our position simple text based information retrieval techniques will more easily enable open-domain narrative constructions due to the scale that can be achieved.

Retrieve

In order to make use of our story corpus, it is necessary to have a mechanism for sentence-level retrieval. In this work, we use the Apache Lucene system (Gospodnetic and Hatcher 2004), a high-performance information retrieval toolkit. Lucene can create positional indexes allowing for fast searches of Boolean, phrase and other query types on a corpus. Lucene also provides a default document ranking function based on a slightly modified version of the popular term frequency-inverse document formula (TF-IDF), which will be described shortly. These features give us the required means for locating similar stories by comparing story sentences, and then selecting the next sentence in retrieved stories as the contribution to the user's developing story.

Lucene creates an index by scanning through every document in a corpus and collecting several key pieces of information about each unique token[1]. Each token stores the total number of documents that contain it. In addition, for each document the token is found, the unique document identifier is stored along with the frequency of occurrence and the location in the document where the token is found. This information can then be used to efficiently find all the documents containing a particular token, such as the token *story*. It can also be used for Boolean queries such as, find all documents with the tokens *story* AND *narrative*, by taking the intersection of the document sets returned. More importantly for our models is the ability to perform phrase queries such as, find all documents that contain the phrase *narrative story*, which can be done using constraints on the positional indexes of the two tokens.

[1] A token in our application is any sequence of characters separated by whitespace after the OpenNLP tokenizer has been applied.

A notion of similarity is also paramount to the meaning of our models. Given a set of documents, obtained using the positional index and a set of query terms, Lucene ranks them using a vector space model and a TF-IDF weighting system. The specific scoring function is:

$$\begin{aligned} score(q,d) &= coord(q,d) \cdot queryNorm(q) \cdot \\ &\sum_{t \in q} \left[tf(t \in d) \cdot idf(t)^2 \cdot queryBoost(t) \cdot norm(t,d) \right] \end{aligned}$$

The key components of this equation are the *tf*, *idf* and *queryBoost* terms. *tf* is the term frequency contribution and is defined as *termFrequency*$^{1/2}$. The more times a term appears in the document the more it contributes to the score. *idf* is the inverse-document frequency and is defined by:

$$idf = \log \frac{totalNumDocuments}{documentFrequency+1}$$

The weight of term is diminished by an increase in the number of documents that contain the term. *queryBoost* is a function that allows you to modify the base weight of the term by a proportional amount. By default this value is 1 and does not affect the overall score of the query. The other terms of the scoring function are for various normalization factors.

Lucene also offers one other feature that we take advantage of in our models. Lucene is able to index documents in a semi-structured way. For example, a single document can contain several, distinct searchable fields. A description of how we utilize this functionality will be introduced later in the paper.

Analyze

The analysis phase in our approach is basically a preprocessing step that modifies input text to be more effective at finding similar stories. For example, consider a story that begins *"Lee knocked on apartment number 534."* If we were to search for sentences like this one, using each word as a query term and Lucene's default scoring mechanism, we would likely receive unsatisfactory results. Recall that TF-IDF gives more weight to terms that appear frequently in a document and reduces the score for terms that appear in many documents. The mechanism has no way of knowing that *knocking* or *being at an apartment* is the semantically important search criteria. In many cases these terms will in fact contribute the most to the overall score, however this example highlights two common problems with this approach. Many proper names and numbers that our users include in their stories are common enough to appear in the database but are uncommon enough that they dominate the TF-IDF scoring function.

To minimize some of these undesired effects we apply the following procedures. First we strip all non ASCII characters. We then apply the Stanford Named Entity Recognizer (Finkel, Grenager and Manning 2005), which we use to replace all matching spans of words with a special token representing either a *Person*, *Organization*, or *Location*. In addition to replacing these entities we also tokenize the input using the OpenNLP toolkit (Baldridge and Morton 2008), lowercase all the text and replace any sequence of numbers with a special token. In order for this preprocessing to work properly, both the text being indexed (the database) as well as the query text (the user's story) need to be preprocessed / analyzed in the same way. Note that these inputs are changed for the purpose of searching the index, but the actual data in the story collection and what will ultimately be returned to the user is unchanged by this process.

Adapt

Adaptation is the process of modifying what has been found in the database and adapting it to fit the user's story more closely. These could be relatively simple things like replacing proper names to match characters of the user's story, or changing the gender or number of the pronouns to reflect the proper relationships in the text. Unfortunately, at this stage of development only a trivial amount of adaptation is applied.

Our approach relies on a sentence detection algorithm, also from the OpenNLP toolkit (Baldridge and Morton 2008), however because weblog text is significantly different than the trained model, more errors than expected still occur. Often sentence fragments are returned that begin in the middle of a sentence or that have trailing garbage after the final punctuation mark. We address these issues by always capitalizing the first character of a returned sentence and by applying a simple heuristic to remove excess trailing characters.

Retrieval Models

Although the architecture is relatively simple and straightforward, several decisions had to be made to build a completely functional system. Primarily these decisions are related to how similarity is defined between story sentences and how this is used to retrieve these sentences from the corpus. As mentioned previously, we use Lucene as the primary basis for answering these questions. Lucene is a powerful toolkit that allows for many query types and optimizations. This leads to a question of what are the best options for finding relevant story sentences, in the database, that will produce the most coherent and entertaining narratives. Swanson and Gordon (2008) showed that with a simple bag-of-words query, that only searches the user's current sentence, can perform with unexpectedly high levels of performance. Although this work showed the viability for such a simple approach, no baselines or comparisons were made. In this section we will discuss five models we believe to be good candidates for search and retrieval of relevant story sentences.

Simple Model

The first model is closely related to the original model proposed in Swanson and Gordon (2008), only differing in the pre and post-processing steps. In this model each sentence in the story collection is treated as an individual entity to be searched (a field). Additionally a pointer to the next sentence is stored along with this entry in the index. In this model the last sentence of a story does not have a searchable field nor does it have a pointer. Queries are constructed using a bag-of-words approach consisting of the words from the user's most recent input sentence only. For all the models, except the baseline, the top ranked sentence returned by the query is displayed to the user as mentioned in the user interface section. The other nine alternatives are populated using the next most similar sentences. See Figure 2a for a representation of how the Lucene index was created corresponding to this model.

Simple Bigram Model

The second model is nearly identical to the *simple model*. The one difference is that we also include bigram phrase queries, along with the simple bag-of-words, in the searching phase.

Context Model

One of the problems discovered in Swanson and Gordon (2008) was the system had no memory. Even when a remarkably similar sentence to the current input was found in the database, often the returned sentence did not make sense with the story as a whole. For example, in just a few turns a story about skiing in Colorado could slip into a day at the beach in Miami. Although the issues involved in tackling this problem broadly are numerous and complex, one solution is to give the system a memory of what has already happened in the emerging story. This can provide a context for the selection of the next event and potentially prevent certain types of incoherencies.

We give our system memory by adding an additional field to our index. When creating our index we crawl through each story one sentence at a time. As usual, we create a document for each sentence that has a field for the current sentence. However, we also create a field, except for the first sentence of a story, consisting of the n sentences prior to the current sentence (in our case $n = 12$). Now when searching we still query the original field with the user's current sentence but we also query the new field with the user's entire story up to the current input.

Despite Lucene's built in normalization procedures that deal with document and query length, a preliminary investigation showed that the context field was still too dominant in the overall score. We attempted to overcome this problem by overriding the *queryBoost* mechanism in the scoring algorithm. Each term in the context query was given a boosted score of $2^{-x/3}$, where x is the number of sentences prior to the current input. This gives less and less weight to terms that appear farther from what is happening in the story now. We have no particular justification for using this function, other than it appeared to do better in our preliminary investigation. A more optimal approach for term weighting will be subject of further research. See

```
Story 1
   Document 1: It was the first time I sailed with Jon Smith.
      Simple Field: it was the first time i sailed with ${person}
      Pointer: Document 2
   Document 2: It was the worst trip ever!
      Simple Field: it was the worst trip ever !
      Pointer: Document 3
   Document 3: I was seasick the whole time.
      Simple Field: i was seasick the whole time .
      Pointer: Document 4
   Document 4: I'll never do that again.
      …
```
a)

```
Story 1
   Document 1: It was the first time I sailed with Jon Smith.
      Simple Field: it was the first time i sailed with ${person} .
      Pointer: Document 2
   Document 2: It was the worst trip ever!
      Simple Field: it was the worst trip ever !
      Context Field: it was the first time i sailed with ${person} .
      Pointer: Document 3
   Document 3: I was seasick the whole time.
      Simple Field: i was seasick the whole time .
      Context Field: it was the first time i sailed with ${person} . it
      was the worst trip ever !
      Pointer: Document 4
   Document 4: I'll never do that again.
      …
```
b)

***Figure 2**: **a)** Illustration of how the index is built for the simple models. **b)** Illustration of how the index is built for the context models.*

Figure 2b for a representation of how the Lucene index was created corresponding to this model.

Context Bigram Model

Like the *simple bigram model* the fourth model uses the same structure as the *context model* but incorporates bigram phrase queries while searching both the simple and context fields.

Random Generation

Although Swanson and Gordon (2008) showed that users generally enjoyed using this type of collaborative writing system and coherent stories could be authored, there was no basis for comparison. The fifth model serves as a baseline by simply returning a random sentence from the database regardless of the user's input.

Examples

In this section we examine three of highly rated stories generated using different models (Figure 3), and highlight some of the qualitative differences between them. Each story is presented in a table in which a row represents one turn of the system. In the first column is the sentence the user wrote (either the first sentence or in response to the computer generated sentence in the previous row). The second column contains the matching content in the database. For the simple (non-context) models this is simply the most similar sentence according the Lucene scoring function. For context models this column is split in two rows. The top row represents the most similar sentence (as in the simple model) and the bottom row represents the preceding *n* sentences of that story.

The first story, presented in Figure 3a, was generated using the *random model*. As expected, in each turn the returned sentence has virtually no overlapping lexical items with the user's sentence. Surprisingly there is an uncanny, coincidental overlap in semantic relatedness of buildings, bodies, etherealness and mortality. Immediately the computer returns somewhat of a non-sequitur (i.e. how do you look out your window if you are outside the building?). Despite this initial problem the user is able to successfully turn the story around and write something other users believe to be coherent and entertaining. This example also brings up two other points. Although the story gets a pretty good rating, it is quite short with the user uninspired to continue beyond 3½ turns. The second thing to note is that these random sentences are actually fairly generic. Although we have not studied this specifically, an informal investigation in the corpus seems to indicate a substantial number of the sentences are fairly content neutral, such as phrases like: "yeah", "oh well", "that's too bad", "I did it last week", "I couldn't believe it". This suggests that a random baseline might not perform as poorly as one would expect. Additionally the *random model* also highlights, what could be considered either a merit or fault depending on your outlook that ultimately much of the quality of the story will be dependent on the creativity and effort exerted by the human user.

The second example, in Figure 3b is a story generated with the *simple bigram model*. Unlike the *random model*, there is considerable lexical overlap between the user's sentence and what is found using the index. For many of the sentences this directly translated into sentences that produced coherent events and actions by the characters in the story. When the lexical overlap corresponds to the semantically relevant parts of the story the computer generated following sentences can be quite convincing, however many times the overlap is on other less relevant aspects of the story. For example the user's phrase, "But we settled for Corfu" shares many bigrams in common with, "But we settled for the simpler name". The problem however is two-fold. Considering just the lexical aspect, despite the significant overlap, the key n-gram that does not align is, "settled for Corfu". The more difficult issues moves beyond lexical relationships and gets at a much deeper problem, that the phrase "settled for Corfu" still does not actually solve the problem. There is nothing in the meaning of "settled for Corfu" that precludes a phrase such as "But we settled for the simpler name". Corfu is indeed a name and with no other information it is still entirely reasonable to expect this sentence to be found again. This highlights a problem of basing the scoring algorithm on lexical frequencies alone and is a fundamental problem in achieving truly coherent and sophisticated stories.

a) Bungie jumping off a building

User written sentence	*Matching sentence*	*Computer generated sentence*
I had never jumped off a building before, until last weekend.		As I was doing that, I suddenly looked out my window.
It was an out of body experience as I watched myself fall.		Oh well.
I thought hoping, no praying, that the harness was tight and the bungie cord was safe.		I was not afraid to leave my earthly body.
Well of course I was a little scared, but if I was not prepared to go all the way I would have never made the first step.		

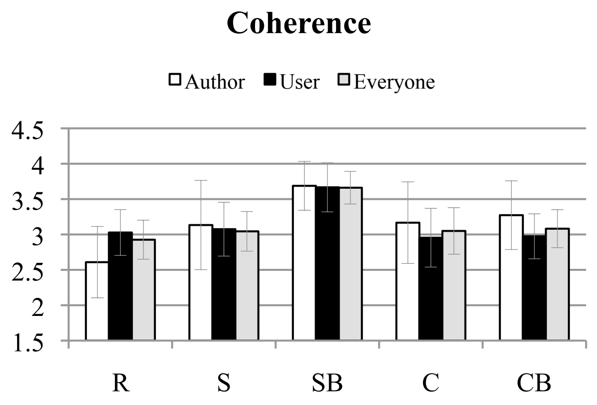
b) Escaping to the island of Corfu

User written sentence	*Matching sentence*	*Computer generated sentence*
It all started easy enough.	It all started out normal enough.	Got up, had lunch, got ready went to work etc.
But then I noticed something very strange.	Then, as I flipped through the test, I realized something very strange.	Mr. Sipiera gave me points on the blank page.
Why would he do that? What was he up to?	Why would he do that?	Because he was planning to kill Sonya for her knowledge of the kidnapping plot.
I decided to get to Sonya first and run away with her to a Greek island.	We left pretty much after this and took a ferry back to Hong Kong island.	Our original plan to go to Lamma island was ditched, as we had inadvertently spent half the day on Lantau island.
But we settled for Corfu, which has a beautiful little hotel on the edge of the old town.	But we settled for the simpler name.	We had dogs (and cats) when I was growing up but my wife and sons have never had a dog before.
Corfu was full of cats and dogs, and I ended up having to look at my feet a lot when walking around town.	and i ended up having to save the day.	It's too long to list the details of the whole operation.
Sonya would take on the secret identity of a scuba instructor, and I'd be a fisherman.	"Before I even got so far as having a proper breakdown in realizing that fantasy is not reality, I found that I had the secret identity of a beetle."	"But that could have been very useful, right?
It meant we could spend a lot of secret time together on the jetty beyond the coastline.	She does spend a lot of time at the bar where they all hang out.	She has told me some of the men she hangs with have come on to her.
I told her I'd kill them just like that old math teacher.	I told her I'd come help her clean and whatever else she needs.	Once she's gone, I'm pretty much on my own.
Fishing all day, looking out for the police.	This is bad for the police.	If they stopped to survey the situation they could only see what was there -- none of the men were armed.
It was a great secret life.		

c) Thunder storm

User written sentence	*Matching sentence*	*Computer generated sentence*
Thank god we put up our lightening rod last weekend!	Thank God we didn't miss the fireworks!	We were near at the Main Building when we already heard explosions in the air.
The largest thuderstorm in 50 years took us by surprise.	Now, being a friendly bunch and people accepting that this was the norm for their works matches, it took us by surprise really.	We all switched to accomodate the extra pegging and with Gary on what looked like a flyer of a peg in front of him, he dug in his heels and said he was not moving.
	We met up with fellow GOT anglers Phil Young (Madd) and Gary Thorpe, the current 5hour record holder. Slowly everyone arrived and although a hard frost had fallen over the previous two night's hopes were still high of a good weight and match. Now after paying our monies to Dave and drawing our pegs should have really told us what sort of day we were to expect.	
I couldn't blame him, it looked like everything not nailed down was going to get blown away.	Don't blame him.	He must be pushing 75.
	They brought back Bebes and he wants to head back to the ranch now. Can't say I blame him. He's no spring chicken anymore." "No coach, they hired Shula." "Shula, well, I'll be," Bryant said. "I never liked him, but he did win a few Super Bowls. So what happened, the old Mount Rushmore face finally hang it up.	
Let's just say calling him stubborn is an understatement.	Foolish is an understatement.	But it's done now.
	I don't need an apology, not any more. Back then -- I might have liked one back then. I was angry for a long time -- at you, at my father, at the system. At me, for being such a fool. Don't say that -- you were never a fool. let a convict talk me into leaving a secured door unlocked, knowing that he was going to escape through it -- that's about as foolish as it gets.	
The storm passed and luckily we weren't all washed away.		

***Figure 3**: **a)** A story generated with the random model (4.29 coherence, 3.71 overall, 7 raters) **b)** A story generated with the simple bigram lucene model (4.22 coherence, 4.44 overall, 9 raters). **c)** A story generated with the context bigram lucene model (3.0 coherence, 2.6 overall, 5 raters).*



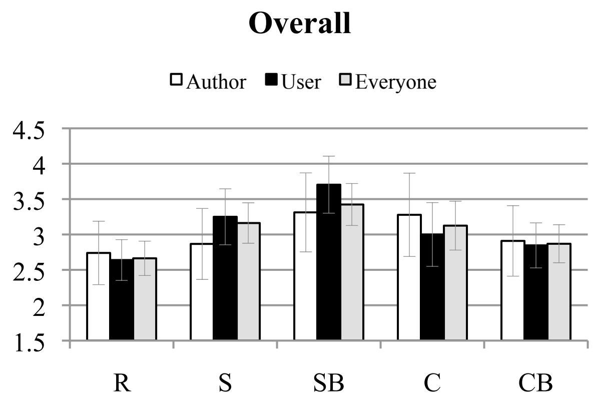


***Figure 4**: Charts comparing the coherence and overall performance of the (R)andom, (S)imple, (SB)Simple Bigram, (C)ontext and (CB)Context Bigram models with error bars representing 95% confidence. The ratings are compared across ratings from the authors only (Author), the users excluding the author (User) and from everyone (Everyone).*

The third example, in Figure 3c, is a story generated using the *context bigram model*. This example basically shows the same benefits and drawbacks as the *simple bigram model*. Despite including the history of the user's story as additional information during the search, it seems the simple field contributes the most dominant weight to the result. It is not entirely clear why this is the case, but at least some of the problem is related to the ad hoc term boosting used to reduce the weight of terms in the history.

Evaluation

The examples presented in the previous section provide some qualitative insight into how the different models behave and what kinds of stories are produced with the system. To make a qualitative comparison we designed a set of experiments that try to assess several different areas of performance. The design of the experiments was similar to those performed in Swanson and Gordon (2008) with a few notable exceptions. When a user began writing a story, one of the five retrieval models was chosen randomly and used for the entire story. When the user was finished with their story they were asked to rate it on the following two criteria using a five-point scale (1 low, 5 high):

Coherence: How much sense can you make out of the story? Do individual sentences follow from each other and does the story make sense as a whole.

Overall: What do you think of the story overall? Did you have fun reading or writing it? Was there anything particularly interesting about it?

Participants in this study consisted of 22 students and staff working at the University of Southern California's Institute for Creative Technologies, recruited by email to use the system over a period of 13 days. Using our web-based interfaces, these users authored a total of 101 stories using a randomly-selected retrieval model, and provided coherence and overall ratings for 96 stories written by other users. In all cases, the authors of these stories remained anonymous to the users who rated them. The results are summarized in figures 4 and 5.

Although, as previously mentioned, much of the success of a story is determined by the creativity of the author, these results show that there are noticeable differences between the models. In general, the *random model* performs the worst on nearly all criteria, while the *simple bigram model* performs among the best. One of the more interesting results, however, is that despite the context models being outperformed on coherence (and overall) ratings by the *simple bigram model*, the authors deemed the choices returned by the system among the best. Another important measure of success for our system is the length of the story, which can loosely indicate the ease and

Model	Coherence	Overall	Combined	Avg. Choice	Avg. Story Length
Random	2.93 ± 0.28	2.66 ± 0.24	2.79 ± 0.18	3.61 ± 0.58	10.43 ± 2.48
Simple	3.04 ± 0.28	3.16 ± 0.29	3.10 ± 0.22	3.00 ± 0.63	11.06 ± 1.85
Simple Bigram	3.66 ± 0.23	3.42 ± 0.30	3.54 ± 0.23	2.55 ± 0.53	14.39 ± 3.46
Context	3.05 ± 0.33	3.13 ± 0.35	3.09 ± 0.28	2.98 ± 0.41	11.89 ± 2.57
Context Bigram	3.08 ± 0.27	2.87 ± 0.27	2.98 ± 0.23	2.68 ± 0.56	10.87 ± 2.78

***Figure 5**: A comparison of several metrics across the models with 95% confidence intervals.* Coherence *and* Overall *are averaged over all the ratings submitted by the users (including the authors).* Combined *is the average of all the* Coherence *and* Overall *ratings.* Avg. Choice *is the average rank of the replacement when the user changes the default selection (given a value of zero).* Avg. Story Length *is the average number of total sentences written by both human and computer.*

enjoyment users have in writing their stories. Again the *random model* performs the worst and the *simple bigram model* performs the best. The role of context is less clear as the *simple context model* produces the second longest stories on average while the *bigram context model* is not much better than the *random model*.

Discussion

Our results show that there are clear differences between the models, however the conclusions that can be drawn are not as decisive. We believed that including context into the search criteria would have improved the quality of the retrieved sentences regardless of our implementation decisions. Unfortunately, the choice to use context is not as simple as a binary decision to include or not include it. It seems it is critically important how much context is included in the search and the weight given to the terms based on factors such as their location. The ability for returned sentences to adhere to the people, states and actions that have already been mentioned in the story is vital to the success of our system. Although our methodology for including context in the retrieval process was not successful we believe that it is still an important avenue to pursue. However, it is also equally important to investigate more advanced adaptation procedures that could also help modify an initially problematic sentence into one that fits with the given narrative constraints.

Although our system relies heavily on information retrieval techniques, this work indirectly indicates that simply adopting the best performing techniques from the IR community may not yield the best results for narrative story generation. Even if we had access to a retrieval system that could find the most semantically similar sentence in the corpus (or a much larger one) there are still several issues that limit the value of these selections in this domain. First, without any adaptation, these sentences (and the ones that follow them) may be appropriately about people doing and saying the right things, but many times they would likely still fail on grammatical issues such as agreement. Also, no matter how large the corpus, names, dates, locations and auxiliary entities are likely to play an even more important role than they would in traditional IR system. However, the more relevant problem to narrative generation is that a good story usually contains events that follow a narrative progression, where events are chosen in support of a developing plotline with an appropriate amount of explication and twists. An ideal system should not always blindly retrieve the most similar sentence from the database, and further research is needed to determine how to do this effectively.

From a high-level view, our approach can be characterized as a type of case-based reasoning system, where new problems are solved by retrieving previously solved problems and then adapting them to the current situation (Riesbeck & Schank, 1989). While our approach takes some inspiration from case-based reasoning, there are still several aspects of our system that could benefit from other core ideas of the case-based reasoning approach. For example, we mentioned the importance of adaptation in returning coherent sentences, but our system still lacks a effective mechanism of transforming a weblog sentence into one that seamlessly integrates into the user's story.

Acknowledgements

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

Baldridge, J., and Morton, T. 2008. The opennlp homepage. http://opennlp.sourceforge.net/index.html.

Cheong, Y.-G., and Young, M. 2006. A computational model of narrative generation for suspense. In *AAAI 2006 Computational Aesthetic Workshop*.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 363–370. Ann Arbor, Michigan: Association for Computational Linguistics.

Gordon, A. S.; Cao, Q.; and Swanson, R. 2007. Automated story capture from internet weblogs. In *Proceedings of the 4th international conference on Knowledge capture*, 167–168. Whistler, BC, Canada: ACM.

Gospodnetic, O., and Hatcher, E. 2004. *Lucene in Action*. Manning Publications.

Mueller, E. T. 2006. Modelling space and time in narratives about restaurants. *Lit Linguist Computing* fql014.

Riedl, M., and Len, C. 2008. Toward vignette-based story generation for drama management systems. In *INTETAIN, Workshop on Integrating Technologies for Interactive Stories*. ACM Digital Library.

Riedl, M. O., and Young, R. M. 2004. An intent-driven planner for multi-agent story generation. *Autonomous Agents and Multiagent Systems, International Joint Conference on* 1:186–193.

Riesbeck, C. K., and Schank, R. C. 1989. *Inside Case-Based Reasoning*. L. Erlbaum Associates Inc. 1

Swanson, R., and Gordon, A. S. 2008. Say anything: A massively collaborative open domain story writing companion. *In First International Conference on Interactive Digital Storytelling*. Erfurt, Germany: First International Conference on Interactive Digital Storytelling.