Social Narrative Adaptation using Crowdsourcing*

Sigal Sina[1], Avi Rosenfeld[2], and Sarit Kraus[1]

1 **Department of Computer Science**
Bar-Ilan University
Ramat-Gan, Israel
sinasi@macs.biu.ac.il, sarit@cs.biu.ac.il

2 **Department of Industrial Engineering**
Jerusalem College of Technology
Jerusalem, Israel
rosenfa@jct.ac.il

Abstract

In this paper we present SNACS, a novel method for creating **S**ocial **N**arratives that can be **A**dapted using information from **C**rowd**s**ourcing. Previous methods for automatic narrative generation require that the primary author explicitly detail nearly all parts of the story, including details about the narrative. This is also the case for narratives within computer games, educational tools and Embodied Conversational Agents (ECA). While such narratives are well written, they clearly require significant time and cost overheads. SNACS is a hybrid narrative generation method that merges partially formed preexisting narratives with new input from crowdsourcing techniques. We compared the automatically generated narratives with those that were created solely by people, and with those that were generated semi-automatically by a state-of-the-art narrative planner. We empirically found that SNACS was effective as people found narratives generated by SNACS to be as realistic and consistent as those manually created by the people or the narrative planner. Yet, the automatically generated narratives were created with much lower time overheads and were significantly more diversified, making them more suitable for many applications.

1998 ACM Subject Classification I.2.1 Applications and Expert Systems

Keywords and phrases Natural language interfaces, Narratives and story generation, Human computer interaction

Digital Object Identifier 10.4230/OASIcs.CMN.2013.238

1 Introduction

Narratives correlate and relate events in our world and represent an important part of human experience. Family members and friends share their experiences via stories. Teachers and leaders tell stories to convey ideas while entertainers tell stories for fun. There is a growing need for conversational agents to understand and validate narratives in settings as diverse as military training, interactive computer games, elderly companion agents and educational, pedagogical and tutoring agents [14, 16, 19, 20, 23, 28, 30].

Due to the significance of this problem, many studies have analyzed storytelling, automatic story generation and interactive narratives. Early works considered a whole story

* This work was supported in part by ERC grant # 267523 and the Google Inter-university center for Electronic Markets and Auctions.

Workshop on Computational Models of Narrative 2013.
Editors: Mark A. Finlayson, Bernhard Fisseni, Benedikt Löwe, and Jan Christoph Meister; pp. 238–256

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

creation without getting any real-time input from the user [16, 28], while later works focus on interactive narratives where the user is part of the story and can affect the plotline [5, 20, 21, 23, 30]. Some previous works use hybrid methods whereby a predefined plot is created and an autonomic agent can later add to or adapt the plot in real-time [19, 22]. However, these studies focus on the general plot of the story but not on the story's details, which were almost exclusively manually created by content experts.

Our goal is to generate realistic narratives as easily and quickly as possible, but with varied content. The narratives we create must be of sufficient quality that they be believable, and thus be useful. Specifically, we present SNACS, an algorithm for generating everyday, social narratives that is customized for a specific storyteller and narrative type. The goal is for SNACS to generate everyday, social narratives that are (a) reasonable (that match the storyteller profile and the input constraints and limitations), (b) consistent (does not contain any contradictions) and (c) realistic (does not raise doubts about the credibility of the storyteller and is compatible with common knowledge implications).

As is the case with several recent works [11, 13, 19, 22], SNACS' algorithm implements a hybrid method which constitutes a "fill and adjust" semi-automatic narrative generation method. However, unique to our work, we add content as follows: We start by generating a dataset of daily, social narratives from Amazon Mechanical Turk workers using a semi-structured form. The dataset contains a collection of these acquired narratives written in natural language along with key attributes from within the narrative and demographic data of the worker. SNACS is a novel algorithm that decides how to use this information dataset in order to output a personalized narrative by adjusting its content according to the requested needs of a specific situation. This paper focuses on describing how the SNACS algorithm operates.

We tested SNACS on four types of social narratives, where the first two types were related to entertainment activities – going out to see a movie or eating at a restaurant, and the other two types were related to errand activities – buying groceries or going to the dry cleaners. We present results that show that SNACS produces narratives which are as reliable, consistent and realistic as the original narratives and a previously defined planning-based approach [17, 18]. Yet our narratives were easier to generate and were judged to be significantly more diversified than the planning-based approach.

2 SNACS Overview, Definitions and Algorithm

We propose and build a system which generates new everyday social narratives using previously stored narratives collected using crowdsourcing techniques. To help clarify and illustrate SNACS' definitions and the algorithm flow, please refer to motivating example 1, a narrative which was generated by SNACS and example 2, the narrative upon which SNACS was based. In example 1, we wish to provide a believable original narrative for a 31-year-old female, who is single and has no children. SNACS generated this narrative based on the narrative in example 2 which was written by a 26-year-old female, who is married and has one child.

▶ **Example 1.** "My sister, my nephew and I went to eat at a restaurant on Friday evening. We went to the nearby city and ate at "Maggiano's Little Italy". This restaurant is one of our favorites so we go there often. Walking in, everything smelled so good and we were greeted at the door and promptly seated since we had called ahead of time. Our waiter was nice and we ordered a nice glass of wine and juice for my nephew. We ordered bruschetta and ordered our main course pasta dishes. My nephew had some mac and cheese. It was very relaxing and the food was amazing. I love going to "Maggiano's Little Italy"."

▶ **Example 2.** "My husband and I went to Olive Garden with our son. This was this past weekend. We went for dinner on Saturday night. We didn't want to cook and wanted to get out of the house. Walking in, everything smelled so good and we were greeted at the door and promptly seated since we had called ahead of time. Our waiter was nice and we ordered a nice glass of wine and juice for my son. We ordered bruschetta and ordered our main course pasta dishes. My son had some mac and cheese. It was very relaxing and the food was amazing. I love going to Olive Garden."

In this section, we describe how this and other narratives can be generated. Note that while both narratives are similar in some ways (e.g., both are narratives about going to restaurants, the lengths of the stories are similar, and both children had mac and cheese), several key differences exist between stories (e.g., one story refers to a person's son, the names of the restaurants are different, and were visited at different times). To explain the process by which some details are retained and others are tailored to the new situation, we begin by providing definitions for SNACS' algorithm. We then describe how information provided through crowdsourcing is used to create a semi-natural language dataset. Last we describe how this dataset can be used to generate a believable narrative to a new storyteller, given her demographic properties.

2.1 Definitions

Throughout this paper, we use the following terms to describe the algorithms that we developed. We notate **original** to refer to elements from the source story upon which we base the new narrative. We notate **generated** for elements related to the narrative that we wish to create. We use examples 1 and 2 to illustrate the definitions and accordingly example 1 is the **generated** narrative while example 2 is the **original** narrative.

Narrative Dataset(DS) – is a collection of narrative records R for a specific narrative type, such as see-a-movie or eat-at-a-restaurant.
Example 2 is an instance of such a record within DS.

Narrative Record (R) – is a triple $\langle P, A, S\rangle$ where: P is the storyteller profile, A is the narrative attributes vector, S is the narrative natural language presentation.
A detailed description of the three fields in R immediately follows. The key to the SNACS' algorithm is how to best select the most appropriate original narrative record from within the entire set of DS. For that every record R is also associated with a tagging vector named ILV (described below), which is the base of the SNACS' selection mechanism. The original narrative record is then modified to create the generated narrative record.

Storyteller Profile (P) – describes the storyteller's properties and consists of gender, age, personal status and number of children.
In examples 1 and 2, the original storyteller profile was: Female, age 26, Married, one child, and the generated storyteller profile was: Female, age 31, Single, no children.

Narrative Attributes Vector (A) – contains a vector of attributes which accompany the narrative. It contains general attributes such as participants, a day, part of day, duration and location. This vector also contains information specific to the narrative domain. It can contain optional values, and thus can be full or partial. Examples of optional values in the restaurant narrative include: location, part of day and participants (e.g., a person can eat at a restaurant alone, but can also go with a spouse, friend or children).

For example, within examples 1 and 2 above, the restaurant narrative attributes are day, part of day, restaurant name, restaurant type, location and participants. Thus, we represent the original narrative attributes vector as: ⟨day (Saturday), part of day (night), restaurant name (Olive Garden), restaurant type (Italian American cuisine), location (downtown) and participants (spouse and son)⟩. The generated narrative attributes vector is: ⟨day (Friday), part of day (evening), restaurant name (Maggiano's Little Italy), restaurant type (Italian American cuisine), location (nearby city) and participants (sister and nephew)⟩.

Natural Language Narrative Presentation (S) – contains a detailed description of an everyday, social activity written in natural language. The natural language (NL) presentation is composed of three parts:

1. The narrative `introduction` – which describes the main facts of the activity, such as who went, when, what are the main object names (which movie/restaurant), where and why.
2. The narrative `body` – which describes the experience in detail, what was the course of events and what happened during the experience.
3. The experience `perception` – which describes how good or bad the experience was from the storyteller's perspective.

We intentionally split the social activity's detailed description into these three parts. This semi-structured natural language writing is very applicable when describing social, everyday situations and and it also centralizes most of the event specific details in the introduction part. This facilitates us to adjust the narrative to a new storyteller profile and attributes vector during the narratives generation.
The original narrative presentation in example 2 is composed of the following three parts:

1. The narrative introduction – "My husband and I went to Olive Garden with our son. This was this past weekend. We went for dinner on Saturday night. We didn't want to cook and wanted to get out of the house."
2. The narrative body – "Walking in, everything smelled so good and we were greeted at the door and promptly seated since we had called ahead of time. Our waiter was nice and we ordered a nice glass of wine and juice for my son. We ordered bruschetta and ordered our main course pasta dishes. My son had some mac and cheese."
3. The experience perception – "It was very relaxing and the food was amazing. I love going to Olive Garden."

Logical Constraints and Common-Sense implications (CS) – contains a set of handwritten, logical rules which the generated narrative should fulfill in order to be reasonable, consistent and realistic. For example:

(a) If the activity occurs late at night, then it shouldn't include small children.
(b) For the see-a-movie activity, the participants should include children if the selected movie type is a children's movie.
(c) The storyteller's profile should match the activity's participants.

Note that we assume that only the generated story need worry about CS, as we assume that the records within DS that were generated through crowdsourcing already fulfill these requirements.

Similarity Attributes Vector (SA) – is a vector of N selected attributes from the storyteller's profile and narrative attributes vector and is defined as $SA =$

$\langle SA_1, \ldots, SA_N \rangle$. SNACS uses this vector to assign values for attributes within a generated narrative as we will explain in this paper.

To generate Social Narratives, SNACS uses a vector of 7 ($N = 7$) known attributes values. These 7 attributes are: gender, age, number of children, personal status, participants, type and part of day. Note that these 7 attributes include all of the profile information attributes (P), but only a subset of the narrative attributes (A).

Similarity Level (SL) – we defined three similarity levels: **same**, **similar** and **other**. We coded these similarity levels as 1, 2 and 3 respectively.

Importance Level Vector (ILV) – a vector ILV contains N values, corresponds to the vector SA, and is defined as $ILV = \langle ILV_1, \ldots, ILV_N \rangle$. Each value in ILV is a value SL and is used to represent the importance of the compatibility of a given attribute SA_i within the narrative body and the experience perception parts of the narrative.

Within SNACS, every record R within the dataset DS has a vector ILV. These values control how much importance should be given to having similarity between the original and generated narratives. Accordingly, if a given attribute within a narrative R can be changed without violating any common sense implications (CS), then the value for SA_i is **other**. At the other extreme, if that attribute is critical and even small variations can make the story implausible, then the value for SA_i is **same**. As we will see in the next section, SNACS considers two ways in which the vector ILV can be built for every record – either a fixed value across all records within DS for a narrative type (which we will call `SNACS-Bst`) or utilizing a content expert to manually tag every record within DS (which we will call `SNACS-Tag`).

The fixed (automatic) ILV contains the **same** value for the gender attribute and a **similar** value for all of the other attributes, i.e., $\langle 1, 2, 2, 2, 2, 2, 2 \rangle$. For the narrative in example 2, the manual ILV is $\langle 3, 3, 3, 3, 2, 1, 2 \rangle$, i.e., $\langle$gender (other), age (other), number of children (other), personal status (other), participants (similar), type (same), part of day (similar)$\rangle$. Note that the type attribute got the **same** value (which means a specific detail) as the narrative contains the sentence "We ordered bruschetta and ordered our main course pasta dishes."

Matching Level Vector (MLV) – a vector MLV contains N values, corresponds to the vector SA, and is defined as $MLV = \langle MLV_1, \ldots, MLV_N \rangle$. Each value in MLV is a value SL and is used to represent the matching level between the original and generated values of a given attribute SA_i. Within SNACS, we built a comparison method for each attribute, which gets as input two values and returns one of the three similarity levels, SL. In case one of the values is missing, it returns the **similar** value, as we assume SNACS will fill this attribute with a similar value. As we will see in the next section, SNACS uses the vector MLV to select the best candidate narrative record R from the DS, in both the `SNACS-Bst` and `SNACS-Tag` variations of the algorithm.

For example, we consider the number-of-children attribute to be the **same** if the difference between the original profile and the generated profile is 1 or less, **similar** if the difference is less than or equal 3 and **other** if one person has children and the other does not or when the difference is greater than 3.

The MLV between examples 1 and 2 will be $\langle 1, 3, 3, 3, 2, 1, 2 \rangle$, i.e., $\langle$gender (same), age (similar), number of children (other), personal status (other), participants (similar), type (same), part of day (similar)$\rangle$.

Compatibility Measure (CM) – The novelty of SNACS is its ability to generate new narratives based on original, acquired narratives. SNACS uses this measure to select the best candidate for the given storyteller profile and the attributes vector.

The compatibility measure CM of a narrative record R, given the vector ILV, a storyteller profile P and a (partial) attributes vector A, is calculated as a weighted sum of scores that depends on the values of the given ILV and the calculated MLV. We describe this calculation later in this section.

The Problem Statement. Given these definitions, we describe how to generate a narrative, S, written in natural language. Narrative S is built from a storyteller profile P, a (partial) vector of attributes A (e.g., time, location and participants) and a set of logical implications CS (data constraints and common knowledge implications). Narrative S must not only be written in natural language but must also be:

(a) reasonable (it matches the storyteller profile and the input attributes),
(b) consistent (does not contain any contradictions) and
(c) realistic (does not raise doubts about the credibility of the storyteller and is compatible with common knowledge implications).

2.2 The SNACS Algorithm

The key to SNACS' success is having a varied set of narrative records within DS from which it can generate new narratives. We used Amazon Mechanical Turk (`http://www.mturk.com/`), a crowdsourcing web service that coordinates the supply and demand of tasks which require human intelligence to create DS. Amazon Mechanical Turk (AMT) has become an important tool for running experiments with human subjects and was established as a viable method for data collection [1, 15]. Our previous experience in running experiments on AMT has demonstrated that this is an effective medium for collecting data about various tasks [2, 3].

We crowdsourced the creation of DS as follows: We built a dedicated, semi-structured questionnaire on AMT to collect the narrative records – the profile, P (gender, age, personal status and number of children), the narrative attributes vector, A (story attributes) and the narrative presentation in natural language S. In this questionnaire the AMT workers were first asked to provide their profiles. Then, the workers were asked to describe daily, social narratives in natural language in as much detail as possible, according to the the three narrative parts – introduction, body and perception. Lastly, the workers were presented with a list of specific questions used to collect the narrative attributes vector, such as "What was the name of the movie/restaurant?", "With whom did you go?" and, "on what day?". The completed record was then stored at the narratives dataset. As we receive new narratives as input from crowdsourcing, we store every record for a given type of narrative (e.g., going to a movie or eating at a restaurant) within the dataset DS_k for that story type. To demonstrate the generality of SNACS, we created four such datasets ($k = 4$) – see-a-movie, eat-at-a-restaurant, buying-groceries and dry-cleaning. We crowsourced 10 narrative records for each narrative type.

Once we wish to generate a new narrative for a listener (user) for a new storyteller profile, we must select the most appropriate narrative record (the original narrative) from DS given that storyteller's profile. We then generate the missing narrative attributes in the narrative attributes vector according to the new storyteller and the chosen record. Finally, we generate the narrative's natural language presentation for the given storyteller profile and the generated attributes vector based on the original chosen narrative's natural language presentation. The process for these steps is formally presented as a SNACS' algorithm and further described later in this section.

Algorithm 1 An Algorithm to create **S**ocial **N**arratives through **A**daptation of **C**rowd-**S**ourcing narratives **(SNACS)**

Require: Storyteller profile P and a (partial) narrative attributes vector A
Require: Logical common knowledge constraints CS
Require: Narrative dataset DS
Ensure: Reasonable, consistent and realistic narrative record R
1: Select original candidate narrative record OR from dataset DS based on P and A
2: Create a new narrative attributes vector NA and complete it according to P, A, CS and OR
3: Create a new narrative record NR from OR, P and NA
4: Replace the original content of the introduction (first part) in NR with a new generated introduction based on P and NA
5: Adjust the second and third parts (the body and perception) in NR
6: **return** The updated NR

The input and output. SNACS gets as input from the listener (user) a storyteller profile P and a vector of optional attributes A (such as participants, or the date). It returns as output a new narrative record NR which contains a reasonable, consistent and realistic narrative presentation S written in natural language.

The algorithm's logical process. The algorithm first selects a candidate narrative record from the dataset (line 1). We present three variations of this selection process below. Then, SNACS completes the missing narrative attributes (line 2). SNACS generates attributes which are similar to the selected, original narrative attributes and matches them to the new storyteller profile. It starts with the participant's generation, who went and how many people participated in the event. It generates the objects' names (movie, restaurant, location) and time frame attributes. For example, if in the original narrative someone went to see a children's movie with his daughter and the new storyteller has no children, the algorithm can choose to include his niece/nephew among the participants. Next, the algorithm generates the narrative's natural language presentation. First, it replaces the original narrative introduction (line 4), i.e., its first part (who went, when, where, why), with a newly generated introduction according to the new profile and the new vector of attributes. This is done by using several predefined Natural Language Generation (NLG) templates with parameters as we describe later in this section.

Finally (line 5), SNACS applies some adjustments to the body and perception parts of the narrative's natural language presentation (the second and third parts). This is done by replacing the references of the original attributes' vector to the new corresponding narrative attributes' vector. In our example, the original participants were a husband and son where the generated participants are sister and nephew; and the original restaurant's name was "Olive Garden" where the generated restaurant's name is "Maggiano's Little Italy". The SNACS algorithm will replace the reference of **My son** with **My nephew** and the reference of **Olive Garden** with **Maggiano's Little Italy** in the following narrative body and perception: "... We ordered bruschetta and ordered our main course pasta dishes. **My son** had some mac and cheese. It was very relaxing and the food was amazing. I love going to **Olive Garden**."

Original narrative selection. We implemented three variations of the SNACS algorithm which differ in how the original candidate narrative record is chosen (at line 1): `SNACS-Any`, `SNACS-Bst`, `SNACS-Tag`.

The `SNACS-Any` variation is a baseline measure that randomly chooses one narrative record from DS. No further logic is performed to check how appropriate that choice is. In contrast, both the `SNACS-Bst` and `SNACS-Tag` variations use a compatibility measure (CM) previously defined. In both of these variations, this measure is used to select which candidate from among all records in DS will serve as the base for the generated narrative. As such, the record with the highest CM value is chosen as the original record.

However, these two variations differ as to how the importance level vector (ILV) is generated. Within the `SNACS-Bst` variation, the algorithm uses the a fixed (automatic) ILV $\langle 1, 2, 2, 2, 2, 2, 2\rangle$ for the CM calculation. In contrast, in the `SNACS-Tag` variation, a content expert manually tags every record within DS to create the vector ILV of the importance of every attributes. Referring back to example 2, the expert tagging generates an ILV of $\langle 1, 3, 3, 3, 2, 1, 2\rangle$. Note that the `SNACS-Tag` variation is much more time consuming than `SNACS-Bst`. As our results will show in Section 4, `SNACS-Bst` and `SNACS-tag` often choose the same record to serve as the base of the narrative. This indicates that we can use the much simpler set values in `SNACS-Bst` and avoid the time of manually tagging the narratives' attributes. Within the `SNACS-Bst` and `SNACS-Tag` algorithms a compatibility measure (CM) is then used to select which record in DS will serve as the original narrative. We use 3×3 matrix scores for all the possible combinations of ILV_i and MLV_i values. For each narrative record R, the algorithm first calculates the MLV for the given storyteller's profile P and input attributes vector A. It then calculates the record's compatibility measure CM as a weighted sum of the corresponding score from the 3×3 matrix based on the ILV (fixed or manual) and the calculated MLV values. Finally, it returns the candidate with the highest compatibility measure for the requested storyteller's profile and the generated narrative attributes vector.

The narrative introduction generation. We generate the narrative introduction (at line 4) using SimpleNLP [9], a Natural Language Generation (NLG), template-based surface realization. Realization is a subtask of NLG, which involves creating an actual text in a human language from a syntactic representation. SimpleNLG is an NLG system that allows the user to specify a sentence by providing its content words and its grammatical roles (such as the subject or verb). SimpleNLG is implemented as a Java library and it allows the user to define flexible templates by using programming variables in the sentence specification. The variable parts of the templates could be filled in with different values. We had a few NLG templates for each narrative type, which were randomly chosen during the introduction generation. For example, two of the NLG templates we use to build a narrative introduction for the see-a-movie narrative are:

(a) Last $\langle time\rangle$ I went to a movie with my $\langle with\rangle$. We went to see the movie $\langle movie\rangle$ at $\langle theater\rangle$.
(b) My $\langle with\rangle$ and I went to see the movie $\langle movie\rangle$ on $\langle time\rangle$. My $\langle with\rangle$ had seen the trailer and wanted to see this movie ever since.

Each template can generate a few variations according to the chosen attributes. For example, the first part of above template (a): "Last $\langle time\rangle$ I went to a movie with my $\langle with\rangle$" where the participants are a wife and son and the time is Sunday afternoon, can generate a few variations, such as:

(a) Last Sunday I went to a movie with my family.
(b) Last weekend I went to a movie with my family.
(c) Last Sunday afternoon I went to a movie with my wife and my son.

3 Comparison with a State-of-the-Art Generator

In order to validate the significance of SNACS, we implemented a planning-based algorithm narrative generator for the see-a-movie and eat-at-a-restaurant narratives. One of the most common approaches for narrative generation is planning-based story generation systems [16, 23, 21, 30, 5]. The planning-based approach uses a causality-driven search to link a series of primitive actions in order to achieve a goal or to perform a task. For the domain of narratives of every day activities, hierarchical scripts can capture common ways to perform the activity. Therefore, we implemented the planner-based generator using a Hierarchical Task Network (HTN). HTN is one of the best-known approaches for modeling expressive planning knowledge for complex environments. It is a natural representation for domains where high-level tasks are decomposed into simpler tasks until a sequence of primitive actions solving the high-level tasks is generated. We used the state-of-the-art SHOP2 planner [18], a well known HTN planner, which has been evaluated and integrated in many real world planning applications and domains including: evacuation planning, fighting forest fires, evaluation of enemy threats and manufacturing processes [17].

Several key similarities and differences exist between generating narratives with SNACS and using any planner, such as SHOP2. First, in both implementations, we assume that the storyteller's profile (P) and a partial vector for A are given. Additionally, we assume that along with P and A, a set of logical constraints (CS) exists that constrains how any narrative can be built, and that Natural Language Generation (NLG) templates will assist with creating the narratives' introduction. However, key differences exist between the Planner and SNACS regrading how the planner will choose the missing narrative attributes and descriptions within the body and the perception of the narrative as we now detail.

Specifying the Planner Input. The HTN planner domain is built according to a **plot graph** of basic actions. A plot graph [31] is a script-like structure, a partial ordering of basic actions/events that defines a space of possible action/event sequences that can unfold during a given situation. For example, the basic actions for see-a-movie include: travel to theater, choose a movie, buy tickets, buy snacks, find seats, see the movie and talk about movie. In our implementation of SHOP2, we manually built the graph plots for two narrative types – see-a-movie and eat-at-a-restaurant. As is the case in SNACS, the planner algorithm starts by filling in the attributes vector (A) according to the logical constraints (described below) and then it searches for a valid plan for the given storyteller profile (P) and the narrative attributes vector (A). As we wanted to get a richer narrative which includes a detailed description of an everyday, social activity written in natural language, we gave the planner an option to tailor natural language descriptions in the basic actions portion of the narrative, as we describe below. Note that in the SNACS algorithm, this step was not necessary as we automatically got the narrative's detailed descriptions from the original narrative. For example, for the "Buy Snacks" action within the see-a-movie narrative type, we gave the planner an option to choose one of the following descriptions:

- We smelled the popcorn and went to buy some.
- We went to get popcorn but it was too expensive.
- We got soda and popcorn from the concession stand.
- We bought some popcorn and drinks and were annoyed at how expensive it was.
- No matter what the price, I just can't see a movie without a big tub of popcorn.
- Snacks at a movie are a complete waste of money! I just bring a snack from home and save the money.

- We decided to buy a large soda to stay cool because the air conditioning wasn't working in the movie theater.
- Though I am supposed to be on a diet, I just couldn't resist buying a few chocolates to eat during the movie.

We defined for each basic action, we defined a set number, 10–15, of different descriptions that were tailored to the specific narrative. These descriptions were manually handwritten by two experts. These experts needed approximately one hour (half an hour per expert) to write the set of descriptions for each basic action option. The experts' data insured that the implemented SHOP2 planner had a variety of descriptions with which to build narratives. Part of these descriptions were also manually tagged by the system's designer with specific tags, such as movie or restaurant types. Note that this tagging is part of the manually supported process needed by this approach in addition to the time spent by the content experts. This tagging gave the generator an option to choose between a generic description which can be associated with any movie/restaurant type or a specific description which can be associated with the current selected movie/restaurant type. For example, for the "Talk about Movie" action we used both generic descriptions, such as:

- It was a nice movie though the end was a bit disappointing.
- I enjoyed the movie though there was a lot of noise from the back seats during the show.
- It was a refreshing and interesting movie even though it was too long for my taste.
- The movie wasn't bad. While I have seen better movies, it was overall a nice way to spend an evening.
- The movie was very nice. I arrived not expecting much and really enjoyed it.

and specific descriptions per movie type, such as

- It was the most stunning film, filled with action and spectacular effects. (action movie)
- I enjoyed the movie. A nice story and very moving. The film was very dynamic, constantly evolving and interesting. (dramatic movie)
- The movie was so funny I couldn't stop laughing. (comedy)
- I highly recommend this film to all ages; whether you are a kid or an adult you will find enjoyment in this film. (children's movie)

The HTN generator flow. The SHOP2-based narrative generator starts filling in the missing attributes vector of the narrative for the given storyteller profile. As the SHOP2-based narrative generator isn't based on an original narrative as in the SNACS algorithm, it uses random selection when there are no logical constraints. The planner implementation first selects the timeframe (when). Next, it chooses the participants, who went and how many people participated in the event, according to the given storyteller profile, the selected time frame and the logical constraints. For example, as in SNACS, if the selected time is night, the narrative planner will not choose children as participants. Lastly, it chooses the objects' names (movie, restaurant, location) and the activity reason (why). In this step, the planner also considers the logical constraints, such as don't choose a children's movie if the participants don't include a child. Once it has all of the activity attributes' values, it looks for a valid the plan according to the domain plot graph. When it has several options from which to choose, it uses a random tie-breaker to select one option, whether it is between different branches of the plot graph or between the basic action descriptions.

The HTN generator output. It is important to note that the output of this planning algorithm is not natural language. The planner algorithm returns as output a raw, semi-structured narrative plan which can been seen (partially) below:

```
(detail action p2 went-to-watch-a-movie)
(detail profile p2 (female 28 married 3))
(detail activity when (4 18 week 3))
(detail activity with kids)
(detail activity name puss-in-boots)
(detail activity loc regal-cinemas)
(detail activity why (my children saw the trailer ...))
(detail action p2 (bought-ticket at-cinema))
(detail action p2 buy-snacks)
(detail action-desc snacks (we bought some popcorn ...))
(detail action p2 found-seat)
(detail action-desc seat (we decided to sit in the first row ...))
(detail action p2 watched-the-movie)
(detail action-desc how (the kids enjoyed the ...))
```

In order to convert the semi-structured plan into a natural language narrative presentation, we implemented a dedicated realizator (SimpleNLP based) which receives this raw, semi-structured narrative plan as an input and returns the narrative presentation written in natural language. This realizator first generates the narrative introduction based on the chosen narrative attributes vector and the storyteller profile using the same NLG templates the SNACS generator used. Next, it generates the event activity descriptions for the body and perceptions parts. For this step, it uses dedicated templates, which we didn't need for the SNACS algorithm as we based these parts on the original narrative that we had in our dataset. These templates were manually predefined for each one of the activity actions. Each template generates the basic description of the action in natural language and combines this output with its' associate chosen description, if one exist in the plan. For instance, the following two lines of the plan: (1) (detail action p2 found-seat) and (2) (detail action-desc seat (we decided to sit in the first row ...)) will generate the following natural language sentences: "We went in to find seats and watched the movie. We decided to sit in the first row because the theater was crowded and I didn't want anyone to block my view."

Overall, the HTN-based narrative generator has an inherently higher cost associated comparing to the SNACS algorithm for the following reasons: Both SNACS and the planner do have the steps of building of the narrative introduction templates and the implementation of the logical constraints. However, the planning-based algorithm implementation also required some additional manual steps. These steps include: the manual building of the plot graph; the writing, associating and tagging of several detailed descriptions for each basic action; and writing a specific realizator for each basic action. Each one of these steps requires both time and resources from a content expert or a system's designer. In fact, because of this cost overhead, we only used the SHOP2 planner in order to define two entertainment narrative types – going out to see a movie or eating at a restaurant. We intentionally did not implement the HTN-based narrative generator for the errand narrative types – buying groceries and dropping off or picking up dry cleaning. Nonetheless, the narratives produced by SNACS were as good as those developed by this costly process, as our results detail in the next section.

4 Experimental Setup and Results

The evaluation of the effectiveness of our narrative generation algorithm SNACS was based on several instances of feedback from Amazon Mechanical Turk participants. The people were presented with narratives generated by our algorithm, the time intensive HTN planning-based generator, the manually handwritten narratives, and the random baselines (describe below).

4.1 Experimental Setup

Specifically, our evaluation was as follows: For each narrative type, we generate 4 storyteller profiles. For each profile, we then generate narratives using both SNACS and the HTN algorithms. As was previously mentioned, we implemented the HTN-based narrative generator only for the entertainment activities – going out to see a movie or eating at a restaurant, and did not implement it for the errand activities – buying groceries and dropping off or picking up dry cleaning, due to the inherent high cost associated with this approach. We also randomly selected four narratives out of a pool of ten original narratives.

The AMT participants were presented with a questionnaire, in which they were asked to grade each of the narratives together with their associated storyteller profiles. The questionnaire contains grades with values from 1 (Least) to 6 (Most). We intentionally chose a scale with an even number of values as we didn't want to allow the users to choose a middle value. The questionnaire asked the participants to grade six aspects of the narratives:

- Authentic – Did you find the story authentic?
- Reasonable – Did you find the story reasonable?
- Profile – Does the story match the storyteller profile (gender, age, personal status, number of children)?
- Coherency – Did you find the story coherent? Does the story make sense overall?
- Fluency – What is the fluency level of the story?
- Grammar – What is the level of the grammar in each sentence?

Each questionnaire contains between 8–10 narratives and was fill out by 8–10 users, to ensure we had a least 30 independent grades per narrative type and narrative generation algorithm. In order to better understand the participants' responses, we also asked them to explain their choices in free text. This ensured that subjects answered truthfully as we were able to manually check these open questions before accepting a participant's grades. Additionally, as we estimate that completing a questionnaire takes 8–12 minutes, we filtered out questionnaires which were filled out within less than 4 minutes as we assumed that the responses were not valid.

Random baselines. We also implemented two random algorithms as baselines to ensure the validity of our experiments. The first random algorithm, to which we will refer as `Rnd-SNACS`, uses the SNACS generator infrastructure. The `Rnd-SNACS` algorithm randomly chooses one of the narratives in the collection and then it randomly chooses the participants, the time frame, location and objects' names. This version ignores the current storyteller profile, the original use profile, the original narrative attributes' vector and the logic constraints and implications. The second random algorithm, to which we will refer as `Rnd-Planner`, uses the planning-based generator. Here, we removed the logical constraints on the participants, time frame and movie or restaurant types from the domain and the problem instance of the original version and use instead random selections to fill in the attributes vector and to choose

Table 1 Average Grades.

	Movie			Restaurant		
Algorithm	**Mean**	**Std**	**N**	**Mean**	**Std**	**N**
Original	4.75758	1.14040	33	4.68981	1.02365	36
Planner	4.52083	1.09639	32	4.25000	1.20780	32
Rnd-Planner	3.71905	1.52097	35	3.35784	1.45998	34
Rnd-SNACS	2.75238	0.90782	35	3.06481	1.17937	36
SNACS-Any	4.47475	1.06484	33	4.30303	0.98817	33
SNACS-Bst	4.42857	1.14567	35	4.52688	1.08277	31
SNACS-Tag	4.43750	0.9482	32	4.46774	1.11589	31

Table 2 Errands Average Grades.

	Buy Groceries			Dry Cleaning		
Algorithm	**Mean**	**Std**	**N**	**Mean**	**Std**	**N**
Original	4.09375	1.25291	32	4.51010	1.23106	33
Rnd-SNACS	3.59896	1.35391	32	3.74479	1.28263	32
SNACS-Any	3.77451	1.39315	34	4.63333	1.21725	35
SNACS-Bst	4.37879	1.45492	33	4.75269	0.94953	31
SNACS-Tag	4.83333	1.00179	32	4.70707	1.15096	33

the detailed descriptions. As the logical constraints and Common-Sense implications (CS) part was removed from these algorithms, we posit that these random baselines will generate poor stories. For instance, within the eat-at-a-restaurant narrative type, the algorithm `Rnd-SNACS` generated a contradiction regarding the time: "On Thursday **night**..." and then later in the story "We went in when the doors opened at **11:00AM** ...". Similarly, in a different generated narrative the `Rnd-Planner` algorithm generated a contradiction regarding how crowded the restaurant was: "We couldn't find a seat and **had to wait** for more than fifteen minutes" yet later in the same story this algorithm generated, "I liked that fact that it was kind of **empty**".

4.2 User Feedback Results

Table 1 shows that our novel SNACS algorithm generated revised story events which were rated by the AMT participants as being as consistent, believable and realistic as the original narratives and those produced by time-intensive planning technique. This is done without the costly overhead involved with manually creating narratives or using a planner-based approach.

Table 1 presents the average grade that AMT participants gave for the entertainment narratives. It is clear that the both of the random variations `Rnd-SNACS` and `Rnd-Planner` got lower grades, between 2.75 and 3.72, which are significantly lower than the generation algorithms and the original narratives' grades (specifically, the ANOVA test for the movie narratives of `Rnd-Planner` compared to `original`, `Planner` and `SNACS-Bst` had a much smaller than 0.05 threshold level with $p = 1.95 \cdot 10^{-4}$, $4.99 \cdot 10^{-3}$ and $9.30 \cdot 10^{-3}$ respectively). These results also show that all non-random generation methods got very similar grades

(4.43–4.47) for the movie narratives and that the `SNACS-Bst` and `SNACS-Tag` grades (4.46–4.52) are slightly better for the restaurant narratives than `SNACS-Any` method, which got an average grade of 4.30. Comparing to the SNACS based generator, the `Planner` generator got slightly better grades for the movie narratives (4.52) and slightly worse grades for the restaurant narratives (4.25). Although the `original` grades, 4.76 and 4.39, for the movie and restaurant narratives are slightly higher than all of the other methods, overall there is no significant difference between all of the SNACS-based algorithms or the planning-based generator or the original narratives.

Table 2 shows the average grades of the errands narratives, Again, the results show that the random variation `Rnd-SNACS` got lower grades for both the buy groceries and dry cleaning activities. For the buy groceries activity, `SNACS-Tag` got a best grade of 4.83 while `SNACS-Bst` got a best grade of 4.75 for the dry cleaning activity. In both cases, there is no significant difference between the grades of `SNACS-Tag`, `SNACS-Bst` and the `original` narratives.

We also tested each grade separately and the results were very similar and can be found in Table 3 and Table 4. Overall, for all four narrative types, there is no significant difference between `SNACS-Tag`, `SNACS-Bst` and the `original` narratives.

Table 3 Stories Aspects Grades.

Narrative Type	Algorithm	AvgGrade	Authentic	Reasonable	Profile	Coherent	Fluency	Grammar
Movie	Original	4.75758	4.97	4.94	5.27	4.88	4.48	4.00
	Planner	4.52083	4.56	4.56	4.59	4.56	4.69	4.16
	Rnd-Planner	3.71905	3.00	3.09	4.09	3.51	4.06	4.57
	Rnd-SNACS	2.75238	2.20	2.40	1.66	3.14	3.37	3.74
	SNACS-Any	4.47475	4.39	4.64	4.36	4.64	4.55	4.27
	SNACS-Bst	4.42857	4.51	4.57	4.14	4.63	4.57	4.14
	SNACS-Tag	4.43750	4.88	5.09	3.84	4.56	4.06	4.19
Restaurant	Original	4.68981	4.72	4.89	4.94	4.53	4.75	4.31
	Planner	4.25000	4.09	4.09	4.06	4.19	4.62	4.44
	Rnd-Planner	3.35784	3.03	3.12	3.62	3.00	3.59	3.79
	Rnd-SNACS	3.06481	2.44	2.42	3.22	3.47	3.44	3.39
	SNACS-Any	4.30303	4.39	4.45	4.42	4.18	4.18	4.18
	SNACS-Bst	4.52688	4.71	4.77	4.48	4.74	4.35	4.10
	SNACS-Tag	4.46744	4.61	4.94	4.65	4.45	4.10	4.06

Table 4 Errands Aspects Grades.

Narrative Type	Algorithm	AvgGrade	Authentic	Reasonable	Profile	Coherent	Fluency	Grammar
Buy Groceries	Original	4.09375	4.06	4.25	4.47	4.22	3.75	3.81
	Rnd-SNACS	3.59896	3.84	3.56	3.47	3.66	3.53	3.53
	SNACS-Any	3.77451	4.03	3.97	3.47	4.03	3.79	3.35
	SNACS-Bst	4.37879	4.64	4.48	4.58	4.30	4.21	4.06
	SNACS-Tag	4.83333	4.87	4.91	4.91	4.84	4.69	4.78
Dry Cleaning	Original	4.51010	4.61	4.64	4.52	4.58	4.39	4.33
	Rnd-SNACS	3.74479	3.63	3.50	3.78	3.84	4.00	3.72
	SNACS-Any	4.63333	4.51	4.49	4.63	4.80	4.57	4.80
	SNACS-Bst	4.75269	5.03	4.94	4.84	4.84	4.58	4.29
	SNACS-Tag	4.70707	4.58	4.61	4.88	4.91	4.58	4.70

4.3 Lexical Variability Results

We assume that different people have different communication types. As a result, stories for different groups will need to use fundamentally different types of language. For example, if we consider a narrative for elderly women, we want the content to focus on activities with grandchildren or a daily exercise routine, whereas if we consider a narrative for a small boy who is bedridden for medical reasons we might instead focus on activities with parents, school or his friends. Besides the essential purpose of generating a good and realistic narrative, we wanted a method which can generate a variety of narratives for different profiles with high lexical variability.

It is possible to objectively measure the level of variability between narratives. The basic lexical variability measure is the Lexical Overlap (LO) method [25, 7] which measures the lexical overlap between sentences, i.e., the cardinality of the set of words occurring in both sentences. Other studies suggest a variation of this method for text documents and produce lexical matching similarity scores which were based on the number of lexical units that occur in both input segments. More recent studies [4, 6, 10] show that adding semantic analysis, such as WordNet [8] and other statistical corpus data, can improve the accuracy of classification of similar documents over the basic methods of lexical matching/overlap.

In our case, we know that the documents are semantically similar as they all describe similar social situations. As such, we only need to measure the narratives' lexical variability and therefore we decided to use lexical matching methods. We chose the Ratio and Cosine models for measuring lexical overlap because recent work found these to be the most effective [12]. The Ratio model [29] is a binary model which only considers the occurrences of the words in each document. The Cosine model [24] is a count model which also considers the number of occurrences of each word within the document. All narrative evaluation scores have been measured without the Stop Words, which is the most commonly preferred method. We use the set of English Stopwords taken from `http://www.textfixer.com/resources/common-english-words.txt`. The similarity scores are defined as follows:

- The Ratio Model – a binary similarity model (Tversky's (1977)[29]):

$$S_{ij}^{\text{Ratio}} = \frac{a_{ij}}{a_{ij} + b_{ij} + c_{ij}}$$

- The Cosine model – a count similarity model (Rorvig's (1999)[24]):

$$S_{ij}^{\text{Cosine}} = \frac{\sum_k x_{ik} x_{jk}}{(\sum_k x_{ik}^2 \sum_k x_{jk}^2)^{\frac{1}{2}}}$$

where:

- x_{ik} counts the number of times that the k-th word occurs in the document i.
- t_{ik} denotes whether the k-th word occurs in the document i, i.e: $t_{ik} = 1$ if $x_{ik} > 0$ and $t_{ik} = 0$ if $x_{ik} = 0$.
- For the i-th and j-th documents, the count $a_{ij} = \sum_k t_{ik} t_{jk}$ is the number of words that are common to both documents.
- For the i-th and j-th documents, the counts $b_{ij} = \sum_k t_{ik}(1-t_{jk})$ and $c_{ij} = \sum_k (1-t_{ik}) t_{jk}$ are the distinctive words that one document has but the other does not.

SNACS' major advantage over hand-crafting stories is the time saved. However, as we note here, SNACS also produced significantly more varied and diversified stories than those based on the planning-based generation algorithm. Table 5 demonstrates this claim by

Table 5 Similarity Scores.

	Movie		Restaurant	
Algorithm	**Ratio**	**Cosine**	**Ratio**	**Cosine**
Original	0.08915	0.35634	0.07192	0.15028
Planner	0.20149	0.55960	0.26066	0.52533
Rnd-Planner	0.19184	0.59161	0.26240	0.51384
RND-SNACS	0.09156	0.40798	0.11768	0.33108
SNACS-Any	0.08417	0.35171	0.12598	0.30366
SNACS-Bst	0.09910	0.51831	0.11548	0.32730
SNACS-Tag	0.10248	0.50804	0.10795	0.32415

presenting the similarity measures, ratio and cosine, for all of the algorithms. As expected, the `original` narratives, which are completely hand-written, got the best scores (lower is better) in both measures. The `original` narratives got a ratio score of 0.089 and 0.072 and a cosine score of 0.36 and 0.15 for the movie and restaurant narratives respectively.

The average ratio score for the SNACS algorithms (`SNACS-Any, SNACS-Bst, SNACS-Tag` and `Rnd-SNACS`) is 0.094 for the movie narratives and 0.117 for the restaurant narratives. The average ratio score for the planning-based algorithms (`Planner` and `Rnd-Planner`) is 0.197 for the movie narratives and 0.262 for the restaurant narratives, which are significant worse than the SNACS algorithms. This difference in the results was seen also in the cosine scores. The average cosine scores are 0.447 and 0.322 for the SNACS algorithms and 0.576 and 0.520 for the planner based algorithms for the movie and restaurant narratives respectively. When looking at the generated narratives, the average ratio scores for all of the SNACS algorithms (`SNACS-Any`, `SNACS-Bst`, `SNACS-Tag` and `Rnd-SNACS`) were significantly lower (which is better) than the planning-based algorithms (`Planner` and `Rnd-Planner`), for the movie narratives and for the restaurant narratives (the ANOVA tests for mean difference of the entertainment narratives' ratio and cosine scores of `Planner` compared to `SNACS-Tag` at the 0.05 significance level being $p = 7.65 \cdot 10^{-4}$ and $2.31 \cdot 10^{-3}$ respectively). Overall, the SNACS algorithms have generated more variable and versatile narratives.

5 Related Work

Studies of automated storytelling have focused on creating systems to generate novel narratives based on prior authored knowledge and logical representations of narrative structure. The two most common approaches to story generation are case-based reasoning and planning. The case-based story generators, such as [28, 27, 11], construct novel narratives by reusing prior narratives, or cases, while planning-based story generation systems, such as [16, 23, 21], use a causality-driven search to link a series of primitive actions in order to achieve a goal. These methods for automatic story generation require that a person primarily author explicitly detail most parts of the narrative, including details about the story and the domain within which it takes place. While such narratives are well written, manually writing makes the development process costly in both time and resources. We compared our automatically generated narratives with those that were planning-based generated and found that overall people found the automatically generated narratives as realistic and consistent as those manually created by content experts. The main advantages of SNACS are the simplicity and the rapidity with which it achieves its results. Furthermore, the narratives

generated with our method are much more varied.

Early studies such as Tale-Spin [16] and Minstrel [28] focused on the creation of a standard representation that can be used to describe characters and their goals, the story's setting, etc. but didn't get any real-time input from the user. More recent studies focus on interactive narrative, which enables the player to make decisions which directly affect the direction and/or outcome of the narrative experience being delivered by the computer system [5, 20, 21, 23, 30]. As the human author is not present at run-time, authoring interactive narratives is often a process of anticipating user actions in different contexts and using computational mechanisms. These studies are similar to ours in that they need to adjust the story to the new constraints presented by the user, but still differ from the current study. They all focus on the creative side of the story and on the high level of the story's plot and characters' goals, while the domain and the story's detailed description were mostly manually written, and we use the wisdom of the crowd to acquire our domain.

Recently, a few studies have started explore ways to automatically build the knowledge base. The SayAnything [26] application constructs new stories from fragments of stories mined from online blogs. The user and computer take turns writing sentences of a fictional narrative, where the sentences contributed by the computer are extracted from Internet weblogs. Our study differs from this work; SNACS generates new content based on the narrative collection while this work only selects and reuses appropriate narrative fragments. In [11] the authors apply case-based reasoning techniques to build an intelligent authoring tool that learns cases from human storytellers who enter dialogue-based narratives via a custom interface. The cases can only be expressed in terms of a known set of possible pre-defined actions. They also use an utterances library which was manually pre-authored and pre-tagged with sets of utterances for each possible dialogue act and therefore limited to a given domain. The authors in [13] introduce an approach to automatically learn script-like sociocultural knowledge from crowdsourced narratives. They describe a semi-automated process by which they query human workers to write natural language narrative examples of a specific, given situation and learn the set of events that can occur and its typical ordering. Our approach also acquires its knowledge from crowdsourced narratives, but it still differs from this work. We use the crowd as the source for the detailed description of situation narratives and thus we focus more on the overall situation and less on the specific, atomic events that compose it.

6 Conclusions and Future Work

In this paper we presented SNACS, a novel approach for generating everyday activities narratives using crowdsourcing. Instead of manually handwriting, which makes the development process costly in both time and resources, SNACS uses crowdsourcing to create a varied base of stories. We compared SNACS to manually handwritten narratives, those generated by SHOP2, a state-of-the-art HTN planner [18] and random baselines. Our evaluation showed that our SNACS narratives were rated as being as believable and consistent as those which are manually handwritten or created from the HTN planner. Yet, the SNACS-based narratives had the main advantage of ease and the rapidity with which these narratives were generated. A second significant advantage to SNACS is that these narratives are rated to be significantly more diversified than those from the HTN planner – all while still maintaining their believability. SNACS is also more adaptable to creating new narratives and it can be easily extended to produce narratives for new domains. In future work we would like to integrate the SNACS algorithm into a dialogue-based application with a virtual human. We

would like to apply our algorithm to different types of applications, such a virtual agent that uses narratives to help interact better with the elderly and for dialogue-based training systems.

References

1 Amazon Mechanical Turk services. `http://www.mturk.com/`.

2 A. Azaria, Y. Aumann, and S. Kraus. Automated strategies for determining rewards for human work. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22–26, 2012, Toronto, Ontario, Canada*, pages 1514–1521, 2012.

3 A. Azaria, Z. Rabinovich, S. Kraus, C. V. Goldman, and O. Tsimhoni. Giving advice to people in path selection problems. In Wiebe van der Hoek, Lin Padgham, Vincent Conitzer, and Michael Winikoff, editors, *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012, Volume 1*, pages 459–466. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

4 D. Bär, T. Zesch, and I. Gurevych. A reflective view on text similarity. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, and Nicolas Nicolov, editors, *Recent Advances in Natural Language Processing, RANLP 2011, 12–14 September, 2011, Hissar, Bulgaria*, pages 515–520, 2011.

5 M. Cavazza, F. Charles, and S. J. Mead. Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4):17–24, 2002.

6 C. Corley and R. Mihalcea. Measuring the semantic similarity of texts. In *EMSEE '05, Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18. Association for Computational Linguistics, 2005.

7 M. Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

8 C. Fellbaum. *WordNet: An electronic lexical database*. MIT Press, 1998.

9 A. Gatt and E. Reiter. SimpleNLG: a realisation engine for practical applications. In Emiel Krahmer and Mariët Theune, editors, *ENLG'09, Proc. of the 12th Europ. Workshop on Natural Language Generation*, pages 90–93. Assoc. for Computational Linguistics, 2009.

10 O. Glickman, E. Shnarch, and I. Dagan. Lexical reference: a semantic matching subtask. In Dan Jurafsky and Éric Gaussier, editors, *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*, pages 172–179. Association for Computational Linguistics, 2006.

11 S. Hajarnis, C. Leber, H. Ai, M. Riedl, and A. Ram. A case base planning approach for dialogue generation in digital movie design. In A. Ram and N. Wiratunga, editors, *Case-Based Reasoning Research and Development*, pages 452–466, 2011.

12 M. D. Lee, B. M. Pincombe, M. B. Welsh, and B. Bara. An empirical evaluation of models of text document similarity. In B. G. Bara, L. Barsalou, and M. Bucciarelli, editors, *Proceedings of the 27th annual meeting of the Cognitive Science Society*, pages 1254–1259. Erlbaum, 2005.

13 B. Li, S. Lee-Urban, D. S. Appling, and M. O. Riedl. Automatically learning to tell stories about social situations from the crowd. In Mark Alan Finlayson, editor, *Proceedings of Computational Models of Narrative 2012*, pages 142–151, İstanbul, 2012.

14 B. Magerko, R. E. Wray, L. S. Holt, and B. Stensrud. Customizing interactive training through individualized content and increased engagement. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Volume 2005: One Team, One Fight, One Training Future*, St. Petersburg, Florida, 2005. Simulation Systems and Applications Inc.

15 W. Mason and S. Suri. Conducting behavioral research on Amazon's Mechanical Turk. *Behavior Research Methods*, 44:1–23, 2012. 10.3758/s13428-011-0124-6.

16 J. R. Meehan. TALE-SPIN, an interactive program that writes stories. In D. R. Reddy, editor, *Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA, August 1977*, pages 91–98. William Kaufmann, 1977.

17 D. S. Nau, T. C. Au, O. Ilghami, U. Kuter, H. Muñoz-Avila, J. W. Murdock, D. Wu, and F. Yaman. Applications of SHOP and SHOP2. *IEEE Intelligent Systems*, 20(2):34–41, 2005.

18 D. S. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

19 J. Niehaus, B. Li, and M. O. Riedl. Automated scenario adaptation in support of intelligent tutoring systems. In R. C. Murray and P. M. McCarthy, editors, *Proc. of the 24th Int'l Florida Artificial Intelligence Research Society Conference*, pages 531–536, 2010.

20 M. O. Riedl. Incorporating authorial intent into generative narrative systems. In Sandy Louchart, Manish Mehta, and David L. Roberts, editors, *Intelligent Narrative Technologies II. Papers from the AAAI Spring Symposium*, number SS-09-06 in AAAI Technical Reports, pages 91–94, Menlo Park, 2009. AAAI Press.

21 M. O. Riedl and A. Stern. Believable agents and intelligent story adaptation for interactive storytelling. In S. Göbel, R. Malkewitz, and I. A. Iurgel, editors, *Technologies for Interactive Digital Storytelling and Entertainment, Third International Conference, TIDSE 2006, Darmstadt, Germany, December 4–6, 2006*, number 4326 in Lecture Notes in Computer Science, pages 1–12. Springer, 2006.

22 M. O. Riedl, A. Stern, and D. M. Dini. Mixing story and simulation in interactive narrative. In *Proc. of the 2nd Artificial Intelligence and Interactive Digital Entertainment Conf., June 20–23, 2006, Marina del Rey, California*, pages 149–150. The AAAI Press, 2006.

23 M. O. Riedl and R. M. Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.

24 M. E. Rorvig. Images of similarity: A visual exploration of optimal similarity metrics and scaling properties of TREC topic-document sets. *Journal of the American Society for Information Science*, 50(8):639–651, 1999.

25 G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of.* Addison-Wesley, 1989.

26 R. Swanson and A. Gordon. Say Anything: A massively collaborative open domain story writing companion. In U. Spierling and N. Szilas, editors, *Interactive Storytelling, First Joint International Conference on Interactive Digital Storytelling, ICIDS 2008, Erfurt, Germany, November 26-29, 2008, Proceedings*, number 5334 in Lecture Notes in Computer Science, pages 32–40, 2008.

27 B. Tearse, N. Wardrip-Fruin, and M. Mateas. Minstrel remixed: Procedurally generating stories. In G. M. Youngblood and V. Bulitko, editors, *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2010, October 11-13, 2010, Stanford, California, USA*, pages 192–197, 2010.

28 S. Turner. MINSTREL: a computer model of creativity and storytelling. Technical Report UCLA-AI-92-04, University of California at Los Angeles, 1993.

29 A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

30 S. G. Ware and R. M. Young. CPOCL: A Narrative Planner Supporting Conflict. In V. Bulitko and M. O. Riedl, editors, *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2011, October 10-14, 2011, Stanford, California, USA*, pages 97–102, 2011.

31 Peter Weyhrauch. *Guiding Interactive Drama.* PhD thesis, Carnegie Mellon University, 1997.