Generating Content for Scenario-Based Serious-Games using CrowdSourcing

Sigal Sina
Bar-Ilan University, Israel
sinasi@macs.biu.ac.il

Avi Rosenfeld
Jerusalem College of Technology, Israel
rosenfa@jct.ac.il

Sarit Kraus
Bar-Ilan University, Israel
sarit@cs.biu.ac.il

Abstract

Scenario-based serious-games have become an important tool for teaching new skills and capabilities. An important factor in the development of such systems is reducing the time and cost overheads in manually creating content for these scenarios. To address this challenge, we present *ScenarioGen*, an automatic method for generating content about everyday activities through combining computer science techniques with the crowd. ScenarioGen uses the crowd in three different ways: to capture a database of scenarios of everyday activities, to generate a database of likely replacements for specific events within that scenario, and to evaluate the resulting scenarios. We evaluated ScenarioGen in 6 different content domains and found that it was consistently rated as coherent and consistent as the originally captured content. We also compared ScenarioGen's content to that created by traditional planning techniques. We found that both methods were equally effective in generating coherent and consistent scenarios, yet ScenarioGen's content was found to be more varied and easier to create.

Introduction

Simulations and scenarios-based games constitute an important subset of serious-games and are an important tool for teaching new skills and capabilities. Such systems are currently used in a broad range of applications such as military, government, educational and health care (Susi, Johannesson, and Backlund 2007; Gandhe et al. 2009; Lane et al. 2010). One main cost in the development of such systems is the time overhead needed by experts to manually create the textual content for these scenarios (Nadolski et al. 2008). While early works considered creating entire scenario narratives without getting any real-time input from the user (Meehan 1977; Turner 1993), later works focused on interactive narratives where the user is part of the story and can affect the plotline (Cavazza, Charles, and Mead 2002; Riedl, Stern, and Dini 2006; Riedl and Stern 2006). Some works also used hybrid methods whereby a predefined plot is created and an autonomic agent can later add to or adapt the plot in real-time (Niehaus, Li, and Riedl 2010). Other work (Riedl and Stern 2006) used a hierarchical decomposition to break a story plan down to primitive actions, however the decomposition rules are manually authored. Overall, these studies focused on the general plot of the story but not on the story's details, which were almost exclusively manually created by content experts. Similar to our work, Li et al. (Li et al. 2012; 2013) used a semi-automated process by which they requested that human workers write natural language narrative examples of a specific, given situation. The workers were requested to segment their narratives such that each sentence contains one event, to use one verb per sentence, and avoid complex linguistic structures. These examples were used to learn plot graphs, i.e. the set of events that can occur and its typical ordering. Our approach also acquires its knowledge from crowdsourced narratives, but significantly differs from this work. While Li et al. used the data acquired from the crowd to build the plot graph, our approach uses a novel method to adjust and personalize the crowd's detailed descriptions for different people within a specific situation in order to generate rich and colorful scenarios.

Research into Procedural Content Generation (PCG) for games has blossomed in recent years and has successfully addressed aspects of content generation in the past decade. However, despite these advances tools for textual content generation are still lacking (Hendrikx et al. 2013). Following Hendrikx et al.'s definition of scenarios, we specifically focus on "concrete game scenarios (that) are explicitly presented in the game, for example as part of the game narrative." Our motivation for focusing on generating this type of everyday content comes from a joint project, **`VirtualSuspect`**, with law enforcement. The project's purpose is to train new detectives to efficiently interview suspects of property felonies. The generative content facilitates ability to add new training cases, allowing investigators to have repeated practice sessions.

This paper's main contribution is ScenarioGen, a general method for generating content which serves as a better alternative to current state of the art traditional planners such as SHOP2 (Nau et al. 2003). ScenarioGen is a powerful algorithm for generating textual content about everyday activities through combining computer science techniques with the crowd. ScenarioGen is composed of three major components – MaxSat (a maximal satisfiability solver), KAR (**K**-nearest neighbor **A**ctivity **R**eplacement) and SNACS (**S**ocial **N**arrative **A**daptation using **C**rowd**S**ourcing). As is

the case with several recent works (Hajarnis et al. 2011; Niehaus, Li, and Riedl 2010; Zook et al. 2012), we implement a hybrid method which constitutes a "fill and adjust" semi-automatic narrative generation method. Accordingly, the MaxSat component identifies the places where modifications are required, the KAR component selects the most appropriate activity for a new scenario and the SNACS component then adjusts the activity adding rich detail. ScenarioGen's novelty lies in how its KAR and SNACS algorithms leverage the crowd to create the textual content from Amazon Mechanical Turk (AMT) workers, an established method for data collection from crowd (Paolacci, Chandler, and Ipeirotis 2010). Using crowdsourcing offers several advantages over previous approaches: First, it enables us to construct the activities' scenarios and narratives' database with less cost and time than traditional planners. Second, it gives us a large and diversified content from the workers, yielding more diversified content than traditional planners. Last, we also use crowdsourcing techniques with another pool of workers, to quickly evaluate and demonstrate the efficacy of our approach in an extensive user study.

While the approach that we present is general and can be used in many scenario-based games, we specifically focused on training scenarios for the joint project **`VirtualSuspect`** with the law enforcement training department (Figure 1). As our approach generates content with relatively low cost and maintenance, we can easily add new training cases, allowing investigators to have repeated practice sessions using different types of investigation techniques for different cases of property felonies. In the **`VirtualSuspect`** project, the generated content can also be applicable as an alibi for a specific case. For example, consider a case where a robber broke into a private house on Sunday night and stole a laptop, jewelry and some money. The law enforcement investigator (the human trainee) can question a suspect, focusing on what he did on Sunday night. The interviewee (our virtual agent), which can be innocent or not, needs a coherent scenario of his Sunday activities that is consistent with the facts that are known to the investigator and/or are common knowledge. While this paper focuses only on describing the interviewee's scenario generation portion of this project, it is important to stress that our proposed approach can also be useful for other applications with everyday content, especially in training scenarios such as training doctors to ask a patient the right questions or to help train candidates in a job interview application.

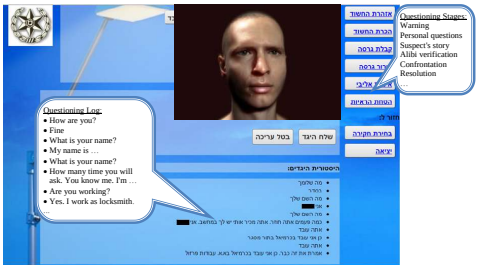

Figure 1: The **`VirtualSuspect`** project

In order to evaluate the effectiveness and generality of ScenarioGen, we implemented and evaluated it to 6 different content domains: two entertainment activities – movie watching and eating out at a restaurant, two errand activities – buying groceries and dry-cleaning, and two types of job descriptions, administrative assistant and technology worker. In all cases we found that the scenarios and their activities' details we generated (examples of which can be seen in examples 1 and 3 below) were rated as being as believable and consistent as the original scenarios and the original activities' details. We also compared the activities' details generated by ScenarioGen to those that were generated with the more costly planning technique and found that ScenarioGen created equally believable content with lower cost and more varied than that of the planner.

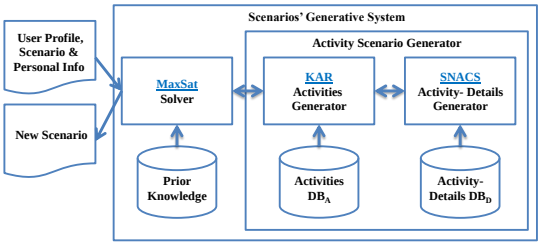

Figure 2: System Modules and Flow

ScenarioGen Description

We proposed and built a system, presented in Figure 2, which generates coherent new scenarios by replacing content details within an original, old scenario. As we analyzed the coherent problem in scenarios, we identified three main questions the system needs to address. These questions are: (1) What should be removed from the scenario? (2) With what should we replace it? (3) How should we replace it? Accordingly, our system consists of three main modules, one to address each of these questions. The modules are: (1) MaxSat, a maximal satisfiability solver that ensures the scenario's integrity and identifies the places where modifications are required, whether it because of an explicit constraint (example 2), or because of implicit constrains, such as distance constrains between places vs. time, which can cause inconsistencies between activities; (2) KAR, an activities generator which uses an algorithm based on the k-nearest neighbor algorithm to choose the most appropriate activity replacement for the required modification; and (3) SNACS, an activity-details generator which adds detail to transform the newly formed content into a realistic and coherent descriptive activity.

To help clarify, examples 2–4 illustrate ScenarioGen's stages. Our system needs to train law enforcement officers to analyze different scenarios. To do so, it creates several variations of John's activity (example 2) by concealing the information that John broke into a house. The MaxSat module will identify the need to replace the $BrokeIntoHouse$ activity. The KAR module suggests as replacement a common activity, such as $EatDinner$ and the SNACS module generates a new activity-details as can be seen in example 3. Our system revises the scenario by basing itself on details from a collection of reported activities, which it modifies to

Example 1: *The following description was generated by our system concerning a 29-year-old single woman who has computer science bachelor degree: "I am a system administrator for a semiconductor firm since May 2011. It is a large sized firm with around 750 employees. The firm provides innovative equipment, services and software to enable the manufacture of advanced semiconductor, flat panel display and solar photovoltaic products. As part of my job, I handle lower level responsibilities like user access and project creations all the way to Site redesigns, release management and upgrades and other IT projects that require management. Overall I love my job. I don't have any complaints."*

Example 2: *John is a 21-year-old male who is single and has no children. He broke into a private house on Sunday night and stole a laptop, jewelry and cash money. He is now being questioning and needs an alibi for Sunday night.*

Example 3: *"On Sunday night I went out for dinner. I did not really want to spend too much so I went to "54th Street". I sat at the bar and got chicken wings. I watched a few basketball games and ate. I read the newspaper a bit too. The food at "54th Street" is always good. The team I picked won so that was also good."*

Example 4: *"I went and got lunch and a beer at a local bar "The Liffey". It was during March Madness, so I was watching some basketball. I sat at the bar and got chicken wings. I watched a few basketball games and ate. I read the newspaper a bit too. The food at "The Liffey" is always good. The team I picked won so that was also good."*

better match the scenario main character's profile. In this example, we base the revised content on a reported activity of a 26-year-old male who goes out to lunch, as can be seen in example 4, and change the details about the time and location to match the required content.

Algorithm 1 presents ScenarioGen's logical flow. ScenarioGen requires as input the user profile, the scenario and a set of constraints which indicates the desired changes or replacement constraints (such as the person's location). It outputs new coherent content where some activities and their descriptions are replaced in order to satisfy the constraints.

Algorithm 1 Scenarios Generator **(ScenarioGen)**

Require: User profile P, original scenario O_Scn and a constraints set S
Ensure: Revise scenario N_Scn
1: Run **MaxSat** solver to get a new scenario N_Scn with AI placeholders
2: **while** N_Scn includes placeholders **do**
3: Run **KAR** to replace a placeholder AI
4: Run **SNACS** to extend the chosen replacement AI into ADR
5: Run **MaxSat** solver to validate N_Scn
6: **end while**
7: **return** N_Scn

We define the following terms for use in the algorithm:

- `User Profile (P)` - describes the user properties (i.e. the scenario's writer or the subject) and consists of gender, age, personal status and number of children.
- `Scenario` - a sequence of activities and their descriptions represented as a list of pairs `<AI, ADR>` where each activity instance `AI` is accompanied with an activity-details record `ADR`. The description of these two fields follows.
- `Activity Instance (AI)` - is a specific occurrence of activity which is part of the scenario and is composed of the activity name, a day, start and end time, location and participants. In our example: $AI(night, John, BrokeIntoHouse, Downtown, alone)$.
- `Activity-Details Record (ADR)` - is a tuple `<P,ADA,ADP>` where: `P` is a user profile, `ADA` is the activity-details attributes vector and `ADP` is the activity natural language presentation. A detailed description of the latter two fields follows immediately.
- `Activity-Details Attributes (ADA)` - contains a vector of attributes which accompanies the activity-details. This vector is a superset of the activity instance `AI`, which contains the general attributes such as participants, a day and location, but it also contains information specific to the activity-details domain, such as restaurant name and type. It can contain optional values, and thus can be full or partial, for example in the eat-at-a-restaurant activity a person can eat at a restaurant alone, but can also go with another person (e.g. a spouse). For example 2, we represent this vector as: ⟨day (Thursday), part-of-day (noon), name (The Liffey), type (Bar and Grill), location (downtown) and participants (alone)⟩.
- `Activity Presentation (ADP)` - is the activity's detailed description written in natural language and is composed of three parts: (1) The activity `Introduction` describes the main facts of the activity, such as who went, when, what are the main objects' names (which movie/restaurant), where and why; (2) The activity `Body` describes the activity in detail, what was the course of events and what happened during the activity; and (3) The activity `Perception` describes how good or bad the experience was from the user's perspective. Note that we intentionally split the activity presentation into these three parts. This semi-structured free text writing is very applicable when describing social, everyday situations. It also centralizes most of the activity specific details in the introduction part, which facilitates adjusting the activity to a new user profile and attributes vector. Accordingly, the presentation of example 2 is: (1) `Introduction`: "I went and got lunch and a beer at a local bar..." (2) `Body`: "I sat at the bar and got chicken wings..." and (3) `Perception`: "The food at "The Liffey" is always good..."

The novelty within the ScenarioGen algorithm is its KAR and SNACS modules and not the MaxSat solver. This is because our use of MaxSat for content generation follows previous work (Biskup and Wiese 2008), and we used the award-winning off-the-shelf "akmaxsat" solver (Kuegel 2010) in our implementation. Both KAR and SNACS use the same similarity measure to find similar activities (in KAR) and content activity details (in SNACS) in the pre-

acquired datasets – the activities dataset, denoted DS_A and the activity-details dataset, denoted as DS_D, respectively (which will be described in details later). Using a comparison function, we compare each pair of attribute values. The function returns one of the 3 values: **same**, **similar** and **other**. Note that in the case one of the attribute values is missing, the function returns the **similar** value as the default value. For example, we consider the number-of-children attribute to be the **same** if the difference between the two values is 1 or less, **similar** if it is less than or equal to 3 and **other** if one person has children and the other does not or when the difference is greater than 3. We now detail how these similarity measures are used in the KAR and SNACS modules.

KAR Activities Generator Module

The activities generator's goal is to find the most appropriate activity replacement, such as $AI(night, John, EatDinner, Downtown, alone)$, for the scenario's placeholder provided by the solver, such as $AI(night, John, PH, Downtown, alone)$. To do this, KAR receives as input a user's profile `P`, the scenario `Scn`, activity records `AR` within the dataset DS_A and the activity placeholder `AI` which needs to be replaced. KAR returns a revised scenario with changed activity which will later be associated with a natural language description and details from the SNACS module. KAR is implemented using the k-nearest neighbor algorithm and it uses likelihood measure (based on the similarity measure described above) to predict which activity record is the best replacement for the placeholder. Specifically, ten attributes are compared in determining activities' similarity: the 4 attributes of the profile `P` and the 6 attributes of the activity `Act`.

The impact of having a high similarity measure can depend on the specific attribute. For example, having the same gender value is likely more important than having the same exact age. We associate different weights for each attribute with a scoring function that gets an attribute and a similarity measure and returns a score within the range [-15,15]. We refine this score function using several preliminary trial and error iterations. KAR calculates the likelihood measure for each of the records, `AR`, as a summation of these scores, then it sorts the activities records according to this measure and uses the k-nearest neighbor algorithm to choose the best candidate. We implemented two variations of this algorithm, one with K=1, which returns the activity with the highest measure and the other with K=11, which also takes into account the number of similar records. As each activity type has different instances in the dataset for different profiles, we gave measures for the specific instances. We keep the top 11 instances with the highest measures and return the activity type which was most common.

SNACS Activity-Details Generator

The activity-details generator module is responsible for adding detail to the revised scenario to make more varied and realistic content. Its input is the user's profile `P`, a partial activity details attributes vector `ADA` (which is made up of the values given in the activity instance `AI`) and the activity-details records `ADR` within the dataset DS_D. It returns as output a new activity-details record `ADR` which contains a consistent and realistic activity presentation `ADP` written in natural language, which substitutes the activity in the revised scenario. The novelty behind SNACS is that its generator is based on the activity-details records collected from the crowd. SNACS first selects the best candidate record (we present 3 variations of this selection process below). Then, it completes the missing activity-details attributes through generating attributes which are similar to the original record's attributes and matches them to the new user's profile. For example, if in the original activity someone went to see a children's movie with his son and the new user has no children, SNACS can choose to include his niece/nephew among the participants. Last, it generates the activity's natural language presentation as follow: (1) It replaces the original activity's introduction, 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, using pre-defined templates. Specifically, we used SimpleNLP (Gatt and Reiter 2009), a Natural Language Generation (NLG), template-based surface realization, which creates an actual text in natural language from a syntactic representation. We created several NLG templates for each activity type, which were randomly chosen. For example, one such template for the movie activity was: "Last $\langle time \rangle$ I went to a movie with my $\langle with \rangle$. We went to see the movie $\langle movie \rangle$ at $\langle theater \rangle$". Each such template can generate a few variations according to the chosen attributes, such as: (a) Last weekend I went to a movie with my family or (b) Last Sunday afternoon I went to a movie with my wife and my son. (2) It adjusts the body and perception (the second and third parts) of the chosen activity by replacing the references of the original attributes' vector with the new corresponding activity, such as the restaurant's name in example 3.

We implemented 3 variations of the SNACS algorithm: `SNACS-Any`, `SNACS-Bst` and `SNACS-Tag`, which differ in how the original candidate activity-details record is chosen. The `SNACS-Any` variation is a baseline measure that randomly chooses one activity-details record from DS_D. In contrast, both the `SNACS-Bst` and `SNACS-Tag` variations use a compatibility measure to select the best candidate from among all records in DS_D. This compatibility measure is based on 7 attributes: the 4 attributes of the profile `P` and 3 attributes from the activity-details attributes vector `ADA` (participants, type and part-of-day). However, when assessing the compatibility of the activity-details record `ADR`, we must also account for the natural language presentation of the activity. Thus, we define an importance level vector `ILV`, which corresponds to these 7 attributes for each record `ADR` within DS_D. Each value in `ILV` is a value `SM` and is used to represent the importance of the compatibility of a given attribute within the activity body and perception parts of the activity presentation. Accordingly, if a given attribute within `ADR` can be modified without violating any common sense implications, then the value is **other**. At the other extreme, if that attribute is critical and even small variations can make the activity implausible, then the

value is **same**. SNACS considers two approaches in which the vector ILV can be built for every record. The first approach, denoted as SNACS-Bst, uses a fixed (automatic) ILV across all records within DS_D. Specifically, it contains the **same** value for the gender attribute and a **similar** value for all of the other attributes. The second approach, denoted as SNACS-Tag, utilizes manually tagging of every record within DS_D. This manual tagging takes less than 2 minutes per description and does not requires technical skills. After reading the description, people grade whether or not the content includes specific details related to the attributes vector. For example, the manual ILV for example 4 is ⟨gender (same), age (similar), number-of-children (other), personal-status (other), participants (other), type (similar), part-of-day (similar)⟩. We again build a score function but this time it gets as input an attribute, a similarity measure and an importance level and returns a score within the range [-15,15]. During runtime, SNACS first calculates the similarity measure of the 7 attributes for all of the records ADR within DS_D compared to the given user's profile P and the (partial) activity instance AI it needs to replace. It then calculates the compatibility measure as a summation of the scores of the 7 attributes based on its calculated similarity measure and its given (fixed or manual) importance level and chooses the record with the highest measure as the best candidate.

Evaluation Methodology

We used the Amazon Mechanical Turk (AMT) platform, an established method for data collection (Paolacci, Chandler, and Ipeirotis 2010), to construct database of possible content replacements with relatively low cost. We also used a second group of crowd-sourced workers to evaluate the efficacy of our approach in an extensive user study (250 subjects). In both populations we ensured that people answered truthfully by including open test questions and reviewing it manually before accepting their work.

We evaluated a total 6 different content domains to judge the generality of ScenarioGen. Motivated by the **VirtualSuspect** application, we studied four different types of scenarios revolving around daily activities: watching movies, eating out at restaurants, buying groceries and going to the dry cleaner. Additionally, to indicate ScenarioGen's usability in other applications, such as content generation for simulating job interviews, we also evaluated how well it could generate content about job descriptions for technology workers and administrative assistants. Examples of all of the questionnaires used to crowdsource the data can be found at http://aimamt.azurewebsites.net/.

Acquiring the Datasets

As described above, both the activities generator (KAR) and activity-details generator (SNACS) rely on large diversified datasets to generate new content based on similarities with existent records. We now describe how we build these two datasets.

KAR Dataset The activities dataset, denoted DS_A, contains a collection of daily schedule records, SR, and activity records, AR. The SR is a tuple <P, Sch> where P is a user profile and Sch is a daily schedule represented as a list of activity instances AI. The AR is a tuple <P, Act> where P is a user profile and Act is the activity properties, defined in a more generic representation than AI, and consists of the activity name and six attributes: a day (a weekday or weekend), part of day, duration, location, participants and frequency. We use two types of forms in order to acquire the two record types. The first form is used to define the set of possible activities and the second form is used to collect additional data on each of the activities from a variety of profiles. In the first questionnaire, we acquire weekday and weekend schedules (in one hour increments), where we asked the workers to describe the activities as specifically as possible and limited each activity to up to a 3 hour duration. For each activity in the schedule, the worker is asked to enter in free text the activity name, the participants and the location. Defining the set of possible activities requires only a few schedules, and thus we collected 16 schedules, 2 schedules from 8 subjects (4 male, 4 female, ages 23-53, paid 25-40 cents). We then map all of the activities in these schedules into an enumerated list and store the converted schedules Sch with their profile P at DS_A as the SR records. The second questionnaire is used to acquire the six attributes of Act for each activity in the collected schedules. The AR records are collected from 60 subjects (23 male, 37 female, ages 21-71, paid 35 cents) which filled-in 12 activities each. Overall we have 720 AR records at DS_A.

SNACS Dataset The activity-details dataset, denoted as DS_D, contains a collection of ADR records which includes: the profile P, the activity attributes vector ADA and the activity presentation in natural language ADP. Here, the workers are asked to describe in free text daily, social activities in as much detail as possible according to the three activity presentation parts - introduction, body and perception. Then, the workers are presented with a list of specific questions. The completed records ADR are then stored at DS_D. We intentionally split the activity's detailed description into three parts for two reasons. First, this semi-structured free text writing is very applicable when describing social, everyday situations, and it helps us to elicit a detailed description of the activity from the workers. Second, it centralizes most of the activity-specific details in the introduction, which allows us to adjust the activity-details to a new user's profile and attributes vector during the activity details generation without the need for intensive usage of NLP tools. We collect and store 10 activity-details for the 4 everyday activities types from 20 subjects (6 male, 14 female, ages 19-55, paid 50 cents) which write two activity-details each. We also collect 32 job descriptions: 16 for positions in computer science field (8 male, 8 female, ages 24-50) and 16 for administrative assistance positions (all female, aged 25-63) which were paid 45-85 cents each. We use similar questionnaire but with a different attributes vector. Overall we have 72 ADR records at DS_D.

Comparison

To evaluate the efficacy of our approach, we compare the generated content to the original handwritten content and to a random based generated content as baseline for verification. In order to validate the significance of SNACS, we

also implemented a second activity-details generator - a traditional planning-based generator described below.

Planner One traditional approach for generating activity-details is to use planning-based systems (Meehan 1977; Ware and Young 2011; Cavazza, Charles, and Mead 2002). The planning-based approach can use a causality-driven search or a hierarchical task network to link a series of primitive actions in order to to perform a task. We based our implementation, denoted as `Planner`, on SHOP2's planning-based generator using a Hierarchical Task Network (HTN) as it is one of the best-known approaches, and has been evaluated and integrated in many real world planning applications such as evacuation planning, evaluation of enemy threats and manufacturing processes (Nau et al. 2003). In conjunction with SHOP2, `Planner` uses a plot graph (Weyhrauch and Bates 1997) which is a script-like structure to partially order basic actions that defines a space of possible action sequences that can unfold during a given situation.

To generate rich content about activity-details including natural language descriptions, we gave `Planner` an option to match natural language descriptions to the basic actions portion of the activity. We defined a set of 10-15 different descriptions that were tailored to each one of the selected actions, which assured the implemented planner had a variety of descriptions with which to build activity-details. These descriptions were manually handwritten and part of them were also manually tagged with specific tags, such as movie or restaurant types. In addition to the introduction's realizator, which we also used in SNACS, we implemented dedicated actions' realizators (SimpleNLP based) that took the planner output, a semi-structured plan, and translated it into a natural language activity presentation. Overall, the HTN-based generator had an inherently higher cost compared to SNACS for the following reasons: Both SNACS and the planner have the steps of building the activity introduction templates and the implementation of the logical constraints. However, the planner implementation also required the following additional manual steps: 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. We implemented this planning-based method only for the movie and restaurant activities because of the cost overhead.

Random Baselines We also implemented 3 random methods as baselines to ensure the validity of the experiments. The first method is a random replacement method, denoted as `Random`, for the activity generator experiment and the other two random methods, `Rnd-SNACS` and `Rnd-Planner`, are for the activity-details generator experiment. `Rnd-SNACS` uses the SNACS infrastructure, where it randomly chooses one of the activity-details records in the dataset and then randomly fills in the rest of the activity attributes. `Rnd-Planner` uses the planning-based generator, where we removed the plan's logical built-in constraints and used random selections instead.

Experiments Setup and Measures

To check the integrity of the scenario's activities list after the replacement of one of its activities, we used the daily schedules we already collected from the crowd for DS_A, as the original scenario (without the activity-details that will be evaluated next). We randomly cut a section of 7-8 hours from each of the original activities list, denoted as `Original` and randomly chose one activity from the list to be replaced. We generated three revised lists by replacing the chosen activity. Two of the lists were generated using `KAR` algorithm, one with `K=1` and the other with `K=11`, and the third used a `Random` replacement. Each activities list was associated with a user profile and a day and the AMT workers were asked to rate 3 aspects: reasonable, matching to profile and coherence from 1 (Least) to 6 (Most). The workers were also asked to try to recognize which, if any, activity was replaced. To evaluate the activity-details generators, for each activity type or job description, we generated 4 user profiles and for each profile generated descriptions for all of the implemented generation methods. We also randomly selected 4 additional descriptions out of the original dataset for each type. Each description was associated with its user profile and we asked a new set of AMT workers to rate 6 aspects of the descriptions: authenticity, reasonable, matching to profile, coherent, fluency and grammar using the same 6-value scale. A total of 250 subjects participated in this evaluation, 80 at the first experiment and 170 at the second (121 male, 129 female, ages 18-72, paid 25-50 cents).

Lexical Variability We assume that different people have different communication types. As a result, descriptions of different groups need to use fundamentally different types of language. Thus, besides the essential purpose of generating a good and realistic descriptions, we wanted a method which can generate a variety of descriptions for different profiles with high lexical variability. The basic lexical variability measure is the Lexical Overlap method (Salton 1989; Damashek 1995) which measures the lexical overlap between sentences, i.e. the cardinality of the set of words occurring in both sentences, while more recent methods also include semantic analysis. In our case, as the documents are semantically similar as they all describe similar social situations, we decided to use lexical matching methods. We chose the Ratio and Cosine models because recent work found these to be the most effective (Lee et al. 2005). The Ratio model (Tversky 1977) is a binary model which only considers the occurrences of the words in each document. The Cosine model (Rorvig 1999) is a count model which also considers the number of occurrences of each word within the document. All evaluation scores have been measured without the Stop Words, which is the most commonly preferred method. The similarity scores are:

$$S_{ij}^{Ratio} = \frac{a_{ij}}{a_{ij}+b_{ij}+c_{ij}}\ ,\ S_{ij}^{Cosine} = \frac{\sum_k x_{ik}x_{jk}}{(\sum_k x_{ik}^2 \sum_k x_{jk}^2)^{\frac{1}{2}}}$$

where x_{ik} counts the number of times the k-th word occurs in the document i, t_{ik} denotes whether the k-th word occurs in the document i, $t_{ik} = 1$ if $x_{ik} > 0$ and $t_{ik} = 0$ if $x_{ik} = 0$. $a_{ij} = \sum_k t_{ik}t_{jk}$ counts the number of common words in the i-th and j-th documents, and 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 doesn't.

Experimental Results

Table 1 shows the results for the first experiment which checks the integrity and coherence of the scenarios activities list after the replacement of one of its activities. It presents the average grades for each aspect and also the total average grade, which was calculated as the average of these three grades. The results show that our generated method KAR K=11 yields the higher results, however, there is no significant difference between the results of Original, KAR K=11 and KAR K=1. As expected the Random variation yields significantly lower results than the others (specifically, the ANOVA test of Random compared to Original, KAR K=11 and KAR K=1 have p-values much smaller than the 0.05 threshold level with p=1.7E-9, 2.8E-11 and 1.1E-8 respectively). It also can be seen from the replacement identification percentage (the last row in Table 1), that only 7% of the users identify the generated activity in the KAR K=11 and KAR K=1 methods which is significantly lower than the 61% in the Random replacement or the 20% in case of an uniform random selection.

	Original	Random	KAR K=1	KAR K=11
Average grade	**4.77**	**3.87**	**4.72**	**4.87**
Reasonable	4.71	3.90	4.68	4.81
Profile	4.56	3.86	4.62	4.65
Coherent	5.05	3.85	4.88	5.16
Replacement identification	---	61%	7%	7%

Table 1: Scenario's Activities List Results

Table 2 shows the results for the second experiment which checks the authenticity, integrity, coherence, matching to profile, fluency and grammar of the generated descriptive activities. It presents the average grade for each activity type and generation method, which was calculated as the average of the six aspects' grades. The results show that our approach produced revised activity-details which were rated as being as coherent and consistent by workers when compared to the original activity-details and the planning-based technique. The main advantage of our approach is that the descriptive activities are much easier to create. Both of the random variations Rnd-SNACS and Rnd-Planner have significantly lower grades than the others (specifically, the ANOVA test for the movie activity-details of Rnd-Planner compared to Original, Planner and SNACS-Bst have p-values much smaller than the 0.05 threshold level with p=1.9E-4, 5.0E-3 and 9.3E-3 respectively). The results also show that all SNACS-based algorithms have very similar grades where SNACS-Bst and SNACS-Tag are usually slightly better than SNACS-Any. Overall for the entertainment activities details, although the Original grades are slightly higher than all of the other methods, there is no significant difference between all of the SNACS-based generator methods or the planning-based generator or the original activity-details. We find similar results for the errands activity-details and for the job descriptions domains where SNACS-Bst or SNACS-tag have the best results, but again there is no significant difference between all of the SNACS-based generator methods or the the original descriptions. As expected the Rnd-SNACS has significantly lower grades than all the others. We find that the results for each of the 6 aspects (omitted due to lack of space) are very similar to the average grade.

Algorithm	Movie	Restaurant	Buy Groceries	Dry Cleaning	Computer Science	Admin Assistant
Original	4.758	4.690	4.094	4.510	4.581	4.581
SNACS-Any	4.475	4.303	3.775	4.633	4.649	4.626
SNACS-Bst	4.429	4.527	4.379	4.753	4.958	4.636
SNACS-Tag	4.438	4.468	4.833	4.707	4.741	4.536
Planner	4.521	4.250	---	---	---	---
Rnd-Planner	3.719	3.358	---	---	---	---
Rnd-SNACS	2.752	3.065	3.599	3.745	3.623	3.876

Table 2: Activity-Details (Narratives) Average Grades

	Movie		Restaurant		Computer Science		Admin Assistant	
Algorithm	**Ratio**	**Cosine**	**Ratio**	**Cosine**	**Ratio**	**Cosine**	**Ratio**	**Cosine**
Original	0.089	0.356	0.072	0.150	0.083	0.241	0.087	0.228
Planner	0.201	0.560	0.261	0.525	---	---	---	---
Rnd-Planner	0.192	0.592	0.262	0.514	---	---	---	---
Rnd-SNACS	0.092	0.408	0.118	0.331	0.077	0.172	0.116	0.222
SNACS-Any	0.084	0.352	0.126	0.304	0.105	0.236	0.062	0.151
SNACS-Bst	0.099	0.518	0.115	0.327	0.097	0.273	0.082	0.146
SNACS-Tag	0.102	0.508	0.108	0.324	0.102	0.261	0.112	0.175

Table 3: Similarity Scores

One major advantage of SNACS over hand-crafting descriptions is the time saved. However, SNACS also produced significantly more varied and diversified stories than those based on the planning-based generation algorithm. Table 3 supports this claim by presenting the similarity measures, ratio and cosine, for all of the algorithms. As expected, the Original descriptions, which are completely hand-written, have the best scores (lower is better) in both measures. The average ratio and cosine scores for the planning-based algorithms (Planner and Rnd-Planner) are significant worse than the SNACS-based algorithms. The ANOVA tests for mean difference of the entertainment descriptions' ratio and cosine scores of Planner compared to SNACS-Tag at the 0.05 significance level yielded p=7.6E-4 and 2.3E-3 respectively. We also calculated these measures for the job descriptions and found that all SNACS-based algorithms generated descriptions which are variable and versatile as the original, hand-written descriptions.

Conclusions

This paper makes the following two key contributions: (i) It is the first work to address the problem of modifying scenarios to generate personal information but yet maintains consistency even when varied scenarios are generated. (ii) It provides a methodology to use crowdsourcing in a principled way for this task. We present ScenarioGen which uses the crowd as the source of our dataset, thus reducing the time and effort needed to generate new scenario content and avoiding the inherent cost of manually modifying scenarios. ScenarioGen uses the MaxSat logical engine in combination with our novel KAR and SNACS modules to generate this content. Our extensive evaluation in 6 different domains shows that revised scenarios and their activities' details derived from ScenarioGen are rated as coherent and consistent as the original scenarios and the original activities' details, yet are generated with significantly lower cost than content created by an accepted planning technique.

Acknowledgments

This work was supported in part by ERC grant # 267523.

References

Biskup, J., and Wiese, L. 2008. Preprocessing for controlled query evaluation with availability policy. *Journal of Computer Security* 16(4):477–494.

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

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

Gandhe, S.; Whitman, N.; Traum, D.; and Artstein, R. 2009. An integrated authoring tool for tactical questioning dialogue systems. In *6th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Gatt, A., and Reiter, E. 2009. SimpleNLG: a realisation engine for practical applications. In *12th European Workshop on Natural Language Generation*, 90–93.

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

Hendrikx, M.; Meijer, S.; Van Der Velden, J.; and Iosup, A. 2013. Procedural content generation for games: a survey. *ACM Trans on Multimedia Computing, Communications, and Applications*.

Kuegel, A. 2010. Improved exact solver for the weighted max-sat problem. *Workshop Pragmatics of SAT*.

Lane, H. C.; Schneider, M.; Michael, S. W.; Albrechtsen, J. S.; and Meissner, C. A. 2010. Virtual humans with secrets: Learning to detect verbal cues to deception. In *Intelligent Tutoring Systems*, 144–154. Springer.

Lee, M.; Pincombe, B.; Welsh, M.; and Bara, B. 2005. An empirical evaluation of models of text document similarity. In *27th annual meeting of the cognitive Science Society*. Lawrence Erlbaum Associates.

Li, B.; Appling, D.; Lee-Urban, S.; and Riedl, M. 2012. Learning sociocultural knowledge via crowdsourced examples. In *Proc. of the 4th AAAI Workshop on Human Computation*.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. 2013. Story generation with crowdsourced plot graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Meehan, J. 1977. Tale-spin, an interactive program that writes stories. In *Fifth International Joint Conference on Artificial Intelligence.*

Nadolski, R. J.; Hummel, H. G.; Van Den Brink, H. J.; Hoefakker, R. E.; Slootmaker, A.; Kurvers, H. J.; and Storm, J. 2008. Emergo: A methodology and toolkit for developing serious games in higher education. *Simulation & Gaming* 39(3):338–352.

Nau, D.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research (JAIR)* 20:379–404.

Niehaus, J.; Li, B.; and Riedl, M. 2010. Automated scenario adaptation in support of intelligent tutoring systems. In *Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*.

Paolacci, G.; Chandler, J.; and Ipeirotis, P. 2010. Running experiments on amazon mechanical turk. *Judgment and Decision Making*.

Riedl, M., and Stern, A. 2006. Believable agents and intelligent story adaptation for interactive storytelling. In *Technologies for Interactive Digital Storytelling and Entertainment*, 1–12. Springer.

Riedl, M.; Stern, A.; and Dini, D. 2006. Mixing story and simulation in interactive narrative. In *Second Artificial Intelligence and Interactive Digital Entertainment Conference*, 149–150. The AAAI Press.

Rorvig, M. E. 1999. Images of similarity: A visual exploration of optimal similarity metrics and scaling properties of trec topic-document sets. *JASIS* 50(8):639–651.

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

Susi, T.; Johannesson, M.; and Backlund, P. 2007. Serious games: An overview.

Turner, S. 1993. MINSTREL: a computer model of creativity and storytelling.

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

Ware, S., and Young, R. 2011. CPOCL: A Narrative Planner Supporting Conflict. In *Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE*.

Weyhrauch, P., and Bates, J. 1997. *Guiding interactive drama*. Carnegie Mellon University.

Zook, A.; Lee-Urban, S.; Riedl, M. O.; Holden, H. K.; Sottilare, R. A.; and Brawner, K. W. 2012. Automated scenario generation: Toward tailored and optimized military training in virtual environments. In *International Conf on the Foundations of Digital Games*. ACM.