Domain Authoring Assistant for Intelligent Virtual Agents

Sepehr Janghorbani
Rutgers University, Disney Research
sepehr.janghorbani@rutgers.edu

Ashutosh Modi
Disney Research
ashutosh.modi@disneyresearch.com

Jakob Buhmann
Disney Research
jakob.buhmann@disneyresearch.com

Mubbasir Kapadia
Rutgers University
mubbasir.kapadia@rutgers.edu

ABSTRACT

Developing intelligent virtual characters has attracted a lot of attention in the recent years. The process of creating such characters often involves a team of creative authors who describe different aspects of the characters in natural language, and planning experts that translate this description into a planning domain. This can be quite challenging as the team of creative authors should diligently define every aspect of the character especially if it contains complex human-like behavior. Also a team of engineers has to manually translate the natural language description of a character's personality into the planning domain knowledge. This can be extremely time and resource demanding and can be an obstacle to author's creativity. The goal of this paper is to introduce an authoring assistant tool to automate the process of domain generation from natural language description of virtual characters, thus bridging between the creative authoring team and the planning domain experts. Moreover, the proposed tool also identifies possible missing information in the domain description and iteratively makes suggestions to the author.

KEYWORDS

Intelligent Virtual Agents, Natural Language Understanding, Text Simplification, Planning Domain Acquisition, Domain Authoring

ACM Reference Format:
Sepehr Janghorbani, Ashutosh Modi, Jakob Buhmann, and Mubbasir Kapadia. 2019. Domain Authoring Assistant for Intelligent Virtual Agents. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Intelligent Virtual Agents (IVA) technology has been applied in many fields such as education [20] or entertainment, where the agents are used for creating virtual user experiences or for digital storytelling [44]. Interactive virtual agent design has been the focus of research for the past two decades [7]. In an effort to give agents deliberative capabilities (a key requirement for social interactions with humans and other agents), a common approach is to use a planner to model the agent's decision making process [24, 37, 46]. Planning architectures are well suited for this problem, since the world of these agents can be modeled as a set of discrete objects, and naturally maps to a logic-based planning domain language. For instance, the world can be modeled by a set of smart-objects [18], each having a set of discrete states and set of affordances [10], where the latter represents the advertised actions of that object.

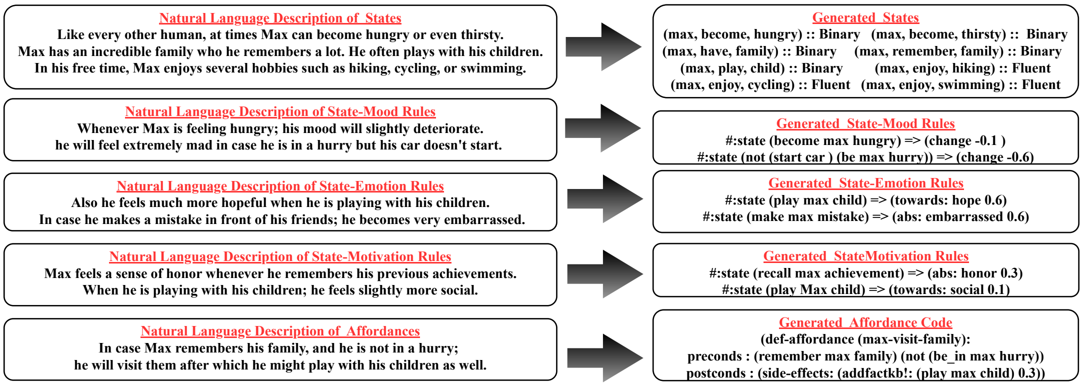
Intelligent agents, especially those designed to exhibit plausible social interactions with human users [4], must exhibit deliberative capabilities, express emotion and personality traits [8], and have mental models of their application domain (e.g., interactive games [13], or storytelling [44]). In order to meet these requirements, creative authors must carefully design aspects of the character's personality and emotion profile, its motivations and its representation of the virtual world it resides in. Creative authors typically use natural language descriptions to design these characters. For example, for describing the emotional profile of Max, (the example character in this paper), a creative would say: "Max becomes slightly angry in case he sees his favorite sports team lose". These natural language descriptions of the agent's behavior need to be translated to a machine-readable planning language, i.e. the domain knowledge of a planner. This translation is manually performed by a team of domain experts, however it is time consuming and resource intensive, and requires frequent interactions between the creative team and the domain experts.

This paper proposes an automated process for translating the natural language description of an agent's behavior to the planning domain knowledge. Authors when describing agent's behavior are typically constrained by domain restrictions, however, the proposed system aims to assist the creative authors in defining, more freely, the behavior of an intelligent character in natural language. The availability of such a tool would not only expand the ease of authoring of the IVAs, but would also promote the applicability of such agents in general, since one of the main bottlenecks in creating IVAs lies in the generation of the agent's domain and its affective states.

The research problem addressed in this paper is inherently challenging due to a number of reasons: (1) non-triviality of the task of automatically understanding natural language, (2) generating an executable planning domain despite the variability in the writing style, and (3) giving the authors as much freedom as possible while still directing them to specify the domain-related material.

Supervised machine learning approaches have been proven to be useful for a similar task of semantic parsing [19]. However, such statistical models cannot be trained for the task at hand due to the lack of sufficient amount of labeled data. In this paper we take an alternate approach for parsing the sentences using dependency graphs [17]. Dependency graphs explicitly describe the syntactic

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.



Natural Language Description of States

Like every other human, at times Max can become hungry or even thirsty.
Max has an incredible family who he remembers a lot. He often plays with his children.
In his free time, Max enjoys several hobbies such as hiking, cycling, or swimming.

Natural Language Description of State-Mood Rules

Whenever Max is feeling hungry; his mood will slightly deteriorate.
he will feel extremely mad in case he is in a hurry but his car doesn't start.

Natural Language Description of State-Emotion Rules

Also he feels much more hopeful when he is playing with his children.
In case he makes a mistake in front of his friends; he becomes very embarrassed.

Natural Language Description of State-Motivation Rules

Max feels a sense of honor whenever he remembers his previous achievements.
When he is playing with his children; he feels slightly more social.

Natural Language Description of Affordances

In case Max remembers his family, and he is not in a hurry;
he will visit them after which he might play with his children as well.

Generated States

```
(max, become, hungry) :: Binary   (max, become, thirsty) ::  Binary
(max, have, family) :: Binary     (max, remember, family) :: Binary
(max, play, child) :: Binary      (max, enjoy, hiking) :: Fluent
(max, enjoy, cycling) :: Fluent   (max, enjoy, swimming) :: Fluent
```

Generated State-Mood Rules

```
#:state (become max hungry) => (change -0.1 )
#:state (not (start car ) (be max hurry)) => (change -0.6)
```

Generated State-Emotion Rules

```
#:state (play max child) => (towards: hope 0.6)
#:state (make max mistake) => (abs: embarrassed 0.6)
```

Generated StateMotivation Rules

```
#:state (recall max achievement) => (abs: honor 0.3)
#:state (play Max child) => (towards: social 0.1)
```

Generated Affordance Code

```
(def-affordance (max-visit-family):
preconds : (remember max family) (not (be_in max hurry))
postconds : (side-effects: (addfactkb!: (play max child) 0.3))
```

Figure 1: Input (left) and output (right) of the proposed model for a small sample domain

relations between different words of the sentence, and hence are useful in extracting information relevant to our task.

While researchers have worked on similar areas of planning domain acquisition, such as robot teaching [21, 23] as well as in the task of story comprehension [5, 15], to the best of our knowledge, we are among the first to address the problem of domain acquisition for developing character-based intelligent virtual agents.

2 RELATED WORK

It has been shown that the natural language description of a planning domain is as expressive as the formal planning domain definition itself [2]. Several researchers have investigated the problem of planning domain generation from natural language text. For example, a highly investigated field of study is automatic planning domain acquisition from natural language instructions directed at robots [9, 21, 23, 33]. These instructions can be translated to the planning domain by taking a rule-based parsing approach similar to ours [23]. Perera and Veloso [33] address the same problem but only parse simple and direct instructions such as “go to” and “deliver” into single actions, and their model does not generalize to a complete domain. Similarly, Pomarlan and Bateman [34] construct an executable robot program using the Cognitive Robot Abstract Machine (CRAM) language, produced by parsing natural language instructions.

The model of Yordanova [45] uses textual instructions for human activities to learn the actions in the planning domain as well as their pre/post-conditions (cf. Section 3). Their method builds an ontology of the domain and then optimizes the model based on sensory data. This approach is suited for the generation of a highly constrained robot planning domain and is less helpful for the highly unconstrained task of intelligent character development.

The generation of action pre-conditions from natural language instructions is addressed by Bindiganavale et al. [1]. Their system works by extracting simple subject-predicate relations in the sentence using an XTAG grammar system. However, the system is suitable for development of simulation environments and not for the task of character development since it gives reasonable performance only in case of simple and direct instructions and does not cover the complete domain. Similarly Goldwasser and Roth [11] train a model to learn a target representation of states and domain rules from a textual description of a simulated environment, e.g., a computer game. They train the parameters of their model with a small number of training data points. Additionally, their model also has a feedback mechanism which generates positive/negative training labels after performing a predicted action. These approaches are suited for development of simulated environments, but for intelligent character development, neither simple instructions nor simulating the agent for feedback generation can be used.

The model proposed in [40] combines the idea of planning domain acquisition with common sense knowledge information, by using semantic role labeling and large-corpus statistics. In our work, we also utilize common sense knowledge, however in a supportive feedback mechanism directed at the authors. The proposed model [40] tries to learn STRIPS (a general-purpose planning domain representation), from web-based data by utilizing common sense knowledge base queries to infer some of the implicit pre/post-conditions.

Another line of work related to intelligent character domain generation is the task of story comprehension [5, 15, 39]. Given a natural language narrative of the story, the goal of this line of research is to understand the key plot of the story as well as the set of events. STAR [5] is an automated story comprehension tool for extracting the key events of a story and doing inference based on the order of the events. It also introduces the concept of a world knowledge in stories, i.e. a set of universal rules governing a wide set of different stories.

Sanghrajka et al. [39] extend STAR by applying a logical reasoning system to detect inconsistencies in the story plot, making inferences about the implicit knowledge in the story. Both systems relate to the concept of story comprehension based on inference from a knowledge-base, similar to the “Common Sense Module” in our system.

Our proposed approach is conceptually similar to StoryFramer [15], which infers the narrative planning domain based on a plot described in natural language. The authors apply linguistic rules on parsed text to connect events with predefined pre/post-conditions. However, our approach differs from StoryFramer as follows: (1) Our system is designed to author the entire domain of an intelligent virtual agent, which includes its inner state, affordances, personality, mood, emotion, and motivations, and a representation of world

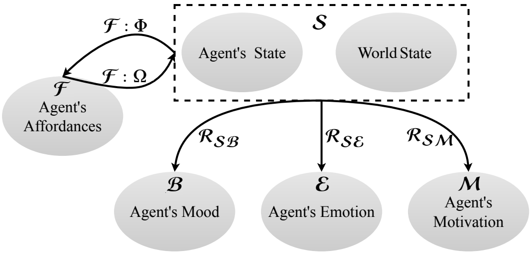


Figure 2: Schematic diagram showing the agent architecture proposed in this paper. It shows the interaction between the different components of the model as described in Section 3.1.

state. This domain significantly builds upon the narrative domain utilized in StoryFramer. (2) Our system supports more complex, compound sentence constructs, which are needed to author the domain described above. (3) Our system supports the automatic extraction of affordance pre- and post-conditions directly from natural language descriptions, thus facilitating generalizability and scalability to completely novel planning domains.

3 AGENT MODEL

In this work, the agent architecture and its world is built around the concept of *smart-objects* [18] where the agent itself is also an active smart-object, and the world describes the set of all smart-objects.

Each smart-object is defined by a set of *states*, representing a set of facts about the smart-object, and *affordances* [10], which reflect the set of offered capabilities on how the smart-object can change its state or the state of other smart objects. Each affordance is defined by a set of pre-conditions and a set of post-conditions.While the state changes are captured in the post-condition of the affordance, its pre-conditions represent the logical conditions on the states which have to be satisfied so that the affordances can be executed.

In our approach, smart-objects with all their states and affordances, including the logic of pre/post-conditions, are inferred directly from a natural language description. Figure 1 shows the input and the generated output for a small example domain. Notice that the post-condition of the affordance is satisfied in a non-deterministic way.

Since the agent architecture is designed for an intelligent character, the agent's state contains not only information about its physical state and the states of the environment, but also the agent's mood, its emotions, and its motivations (Figure 2). While emotions represent short term variables of affect, the mood is a long term concept shaping agent's behavior. The motivation states are used to build the objective function that is used by the planner - in our case a heuristic based search algorithm - that generates plans bringing the agent closer to a set of target motivation values. All three sets of variables are influenced by state-dependent rules, as symbolized by the arrows in Figure 2.

Although our implementation focuses on this particular agent architecture, the methods developed in this paper are fairly general and can be extended to other agent architectures [12, 32], or planning domains (e.g., PDDL) as well. For instance, when the behavior should be influenced by social rules, when the utility of the agent's goal is not encoded with motivations but another objective, or when the affective state are modeled with other mechanisms (or not at all), natural language can be used to build the domain and facilitate the authoring process. Future research will focus on making the approach more general to a broader class of agent architectures in this field.

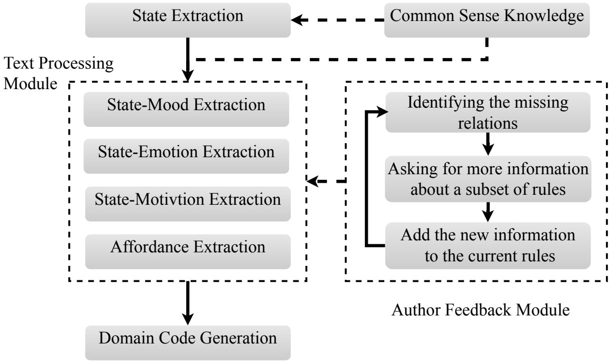


Figure 3: Pipeline for going from natural language description to domain code, as described in Section 4. Solid arrows represent actual data transfer while dashed arrows indicate giving suggestions to the author.

3.1 Problem Domain Definition

Formally, the agent is defined by a tuple $\Sigma = \langle \mathcal{S}, \mathcal{B}, \mathcal{E}, \mathcal{M}, \mathcal{F} \rangle$, with $\mathcal{S}$ representing the state space, $\mathcal{B}$ the agent's mood, $\mathcal{E}$ the agent's set of emotions, $\mathcal{M}$ its motivations, and $\mathcal{F}$ the set of affordances. Each affordance $f \in \mathcal{F} : f = \langle O, \Phi, \Omega, \Lambda \rangle$ is itself a tuple of the affordance owner O, a set of pre-conditions Φ, a set of post-conditions Ω, and side effects Λ.

- $\mathcal{S}$ represents the set of agent's states and the state of the world, see Figure 2. The model differentiates between binary states (e.g. whether the agent is asleep or not) and n-ary states (e.g. agent's position), also referred to as *fluents*.
- $\mathcal{B}$ represents the agent's mood. It is a continuous variable in the range $[-1, 1]$, where -1 represents a negative mood, 0 a casual mood, and 1 a positive mood.
- $\mathcal{E}$ represents the current emotion the agent is experiencing. The emotions are modeled using the Pleasure, Arousal, Dominance (PAD) formalism [26] which associates to each emotion a position in the 3-dimensional PAD space [8].
- $\mathcal{M}$ represents motivations which drive agent's decision making system. The motivations are modeled using Reiss Motivational Profiles (RMP) [36] covering the full range of motivations into a set of 16 motivational factors, each factor being a number in the range $[0, 1]$. Examples of such motivations include "honor", "order" and "family-relationships", etc.
- $\mathcal{F}$ is the agent's set of affordances. These represent the set of actions that can be performed to change the states of one or more smart-objects. An affordance can be applied when a logical conjunction of states (i.e. pre-conditions Φ) is satisfied, and upon successful execution it changes states

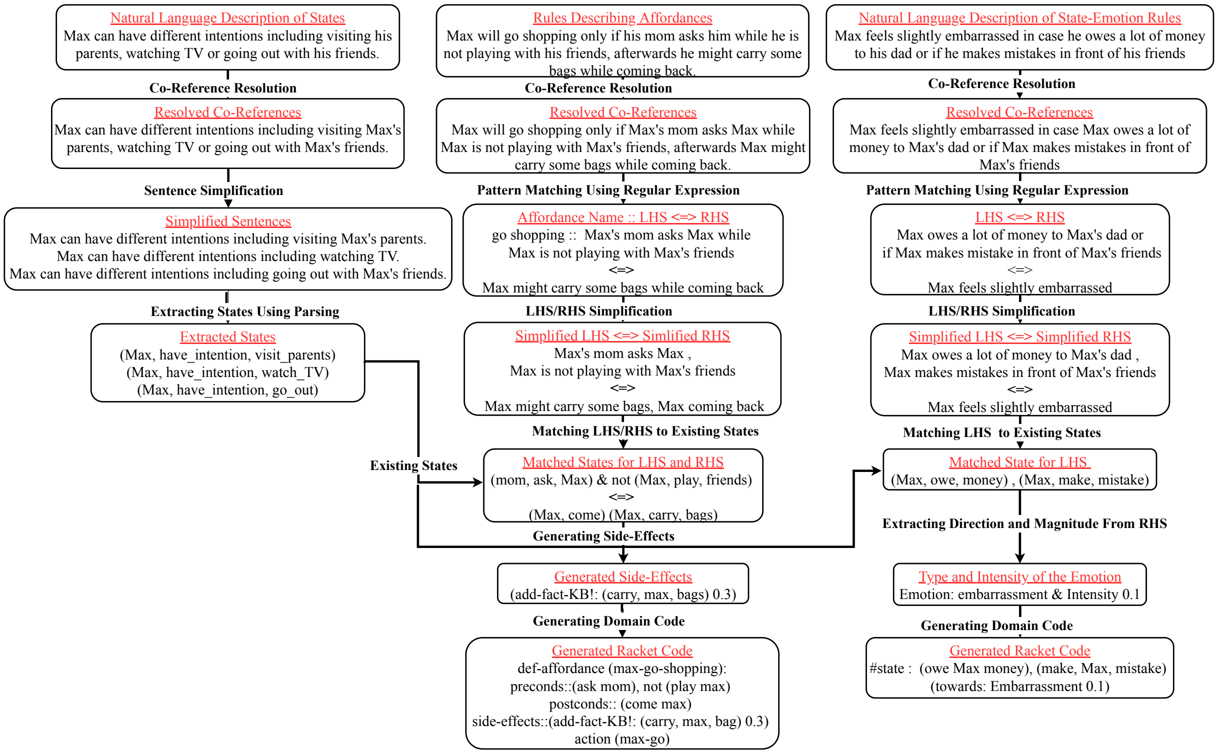

Natural Language Description of States
Max can have different intentions including visiting his parents, watching TV or going out with his friends.
Co-Reference Resolution
Resolved Co-References
Max can have different intentions including visiting Max's parents, watching TV or going out with Max's friends.
Sentence Simplification
Simplified Sentences
Max can have different intentions including visiting Max's parents.
Max can have different intentions including watching TV.
Max can have different intentions including going out with Max's friends.
Extracting States Using Parsing
Extracted States
(Max, have_intention, visit_parents)
(Max, have_intention, watch_TV)
(Max, have_intention, go_out)
Existing States
Rules Describing Affordances
Max will go shopping only if his mom asks him while he is not playing with his friends, afterwards he might carry some bags while coming back.
Co-Reference Resolution
Resolved Co-References
Max will go shopping only if Max's mom asks Max while Max is not playing with Max's friends, afterwards Max might carry some bags while coming back.
Pattern Matching Using Regular Expression
Affordance Name :: LHS <=> RHS
go shopping :: Max's mom asks Max while Max is not playing with Max's friends
<=>
Max might carry some bags while coming back
LHS/RHS Simplification
Simplified LHS <=> Simlified RHS
Max's mom asks Max ,
Max is not playing with Max's friends
<=>
Max might carry some bags, Max coming back
Matching LHS/RHS to Existing States
Matched States for LHS and RHS
(mom, ask, Max) & not (Max, play, friends)
<=>
(Max, come) (Max, carry, bags)
Generating Side-Effects
Generated Side-Effects
(add-fact-KB!: (carry, max, bags) 0.3)
Generating Domain Code
Generated Racket Code
def-affordance (max-go-shopping):
preconds::(ask mom), not (play max)
postconds:: (come max)
side-effects::(add-fact-KB!: (carry, max, bag) 0.3)
action (max-go)
Natural Language Description of State-Emotion Rules
Max feels slightly embarrassed in case he owes a lot of money to his dad or if he makes mistakes in front of his friends
Co-Reference Resolution
Resolved Co-References
Max feels slightly embarrassed in case Max owes a lot of money to Max's dad or if Max makes mistakes in front of Max's friends
Pattern Matching Using Regular Expression
LHS <=> RHS
Max owes a lot of money to Max's dad or if Max makes mistake in front of Max's friends
<=>
Max feels slightly embarrassed
LHS/RHS Simplification
Simplified LHS <=> Simplified RHS
Max owes a lot of money to Max's dad ,
Max makes mistakes in front of Max's friends
<=>
Max feels slightly embarrassed
Matching LHS to Existing States
Matched State for LHS
(Max, owe, money) , (Max, make, mistake)
Extracting Direction and Magnitude From RHS
Type and Intensity of the Emotion
Emotion: embarrassment & Intensity 0.1
Generating Domain Code
Generated Racket Code
#state : (owe Max money), (make, Max, mistake)
(towards: Embarrassment 0.1)

Figure 4: Detailed pipeline with examples for each submodule.

according to its post-conditions Ω. Additionally, the affordances in this particular agent architecture may contain a set of so-called side-effects, Λ, which allow the affordance to change states in a non-deterministic way.

The agent's architecture contains several mechanisms to model the changes in the agent's affective states (emotions, mood, and motivations). These mechanisms are rule based functions $\mathcal{R}(\mathcal{S},\mathcal{A}) : \mathcal{S}_f \rightarrow \mathcal{A}$ from states to affects, where $\mathcal{S}_f$ is a conjunctive normal form (CNF) on states $\mathcal{S}$ and $\mathcal{A} \in \{\mathcal{B},\mathcal{E},\mathcal{M}\}$.

In this paper, we interchangeably use the terms pre-condition and left hand side (LHS) and similarly the terms right hand side (RHS) and post-condition. The rule-based functions are described below:

State-Emotion Rules $\mathcal{R}(\mathcal{S},\mathcal{E})$ address the changes of the emotions, the position in PAD space, based on the current state. The RHS reflects a strength and direction of the shift of the PAD values towards an emotion (e.g increase 0.2). See the right column of Figure 4.

State-Mood Rules $\mathcal{R}(\mathcal{S},\mathcal{B})$ describe how mood changes given the current states. While the LHS is a conjunctive normal form on the state space, the RHS specifies the direction and step size of the mood change.

State-Motivation Rules $\mathcal{R}(\mathcal{S},\mathcal{M})$ represent the mapping from states to the motivations. The RHS either specifies the direction and the strength of change towards any of the motivations or it directly sets the motivations to a specific value.

This paper proposes a model for generating $\langle \mathcal{S},\mathcal{F},\mathcal{R}_{\mathcal{SB}},\mathcal{R}_{\mathcal{SE}},\mathcal{R}_{\mathcal{SM}}\rangle$, given a natural language description of the domain provided by a creative author.

We assume that the set of possible affective states $(\mathcal{B},\mathcal{E},\mathcal{M})$ are predefined and known by the authors. Our proposed system must guarantee that it finds a consistent set of states and is able to use it for generating the logical components the planning system relies on. This effort involves multiple challenges as discussed in the following sections.

4 DOMAIN GENERATION FROM TEXT

For generating domain code from natural language descriptions we propose a modular system, as shown in Figure 3. The first step in the system pipeline is the *State Extraction Module*, which is used to identify and extract the state-related information from each sentence, and subsequently used to construct the states of the smart-objects as well as the states of the world $\mathcal{S}$. Next, the *Text Processing Module* extracts the affordances as well as the state-affect rules using similar mechanisms (see Section 4.1). Lastly, the *Domain Code Generation Module* generates a computer readable planning language from states, affordances and state-affect rules.

Figure 4 exemplifies the entire pipeline. Since the mechanisms used in all state-affect extraction modules are very similar, only the state-emotion extraction is shown.

For the agent architecture in this paper, Racket [6] was used as the target language, however, generalization to other general purpose planning languages (e.g. PDDL) is straightforward [25].

In addition to the processing steps described above, we introduce two more modules that significantly help in generating domain from natural language: (1) a *Common Sense Knowledge Module* and (2) an *Author Feedback Module*. The common sense knowledge module (cf. Section 4.5) facilitates the authoring by suggesting new information about affordances or new rules regarding affective states. For these suggestions a common sense knowledge-base is queried, i.e., ConceptNet [41]. Besides finding missing information, this module should also promote the creativity of the authors by providing new and relevant information about the already authored objects. Furthermore, in case of an unclear natural language description, the author feedback module (cf. Section 4.4) can inform the creative author of possibly missing information about the domain, such as incomplete affordances or possible state dependencies.

4.1 State Extraction

Natural language text can be ambiguous w.r.t. entities mentioned in the text. To resolve these ambiguities about the entity references (e.g., entity referred by a pronoun) we perform co-reference resolution. For example, in the sentence “Max brings the book and then *he* reads it.”, from the point of computational language processing, it is ambiguous whether “he” refers to Max or the book. The co-reference module helps to resolve it. The normalized sentence after the co-reference resolution is “Max brings the book and then *Max* reads it.’. The co-reference normalized sentences are further simplified using rules based on lexical substitution and syntactic reduction techniques [38]. These rules reduce a complex sentence into multiple simpler sentences each with a single main verb. In order to extract state related information, each of the simple sentences is then parsed using a dependency parser [16]. Dependency parser processes a sentence to produce a dependency graph which gives syntactic relations (e.g., subj, obj, etc.) between the words in the sentence. Figure 5 shows an example of a dependency graph obtained for an example sentence. Linguistic simplification rules are then applied on the dependency graph to extract information used for constructing the states. These linguistic rules were designed based on several sample writing styles provided by professional creative authors, and we tried to keep the rules as general as possible. We describe these rules in the following paragraphs.

Before the text is simplified based on linguistic rules, types of states mentioned in the sentences are identified, i.e. either fluent (“*n*-ary”) or “binary”. This identification is done by checking the presence of certain keywords in the sentence that are generally used to describe fluent variables. Examples of keywords used are “including, such as, consist of”, etc. The sentences are then processed with different sets of linguistic rules depending on the state type.

Since a sentence corresponding to a fluent state has one of the mentioned keywords, it typically follows a specific syntactic structureGenerally, the mentioned keywords in sentences containing a fluent, syntactically occur as the prepositional modifier and have prepositional complement (pcomp) dependency relationship with the next word, which is usually a content-rich word carrying important information. For such sentences, words corresponding to the syntactic relations such as prepositional modifier, prepositional complement and their object are extracted. An example of each of these relations as well as the extracted states is provided in Table 1. The extracted state is constructed with a triplet: subject, verb (+ object) and prepositional complement (+ object). Contrary to sentences with fluents, sentences with binary states can have more diverse syntactic patterns. This requires extraction of different and more diverse set of syntactic relations. For such sentences, words corresponding to the relations such as prepositional modifier, adjectival complement, open clausal complement alongside their objects (see Table 1) are extracted.

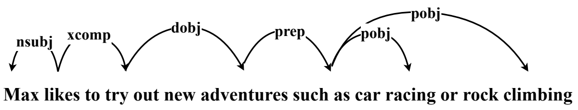

nsubj
xcomp
dobj
prep
pobj
pobj
Max likes to try out new adventures such as car racing or rock climbing

Figure 5: Dependency parse graph for a given sentence from which underlying states: (Max, try_out, racing) and (Max, try_out, climbing) are extracted. The abbreviations used in the figure are as follows: (nsubj: noun subject, xcomp: open clausal complement, dobj: direct object, prep: prepositional modifier, pobj: object of preposition).

In Table 1, the first two rows show examples of sentences with a fluent while the rest are sentences containing binary states. The linguistic rules shown here serve as illustrative examples, and can easily be extended to be more flexible, or accommodate a wider and more diverse range of writing styles.

4.2 Affordance Extraction

To identify the RHS and LHS of the affordance descriptions, the co-reference normalized text is processed with a pattern matching algorithm. These diverse sets of patterns have also been defined based on a creative author’s recommendation for natural writing style and provide flexibility to the authors in describing affordances. For example, in the sentence “Max goes to the library *only if* he has an exam *after which* he feels more knowledgeable.”, the keywords *only if* and *after which* are separating different parts of the affordance. The part before the first keyword is used for obtaining the name of the affordance while the next two parts constitute the set of pre-conditions and post-conditions respectively. Similarly, other patterns are also defined to obtain the pre/post-conditions and to generalize over different language formulations. These patterns can be easily extended or customized.

After extracting the LHS and RHS using the pattern matcher and subsequently simplifying LHS/RHS, the rule-based parsing approaches from Section 4.1 are used to extract the states. Next, the extracted state names are mapped to the already existing states obtained from the State Extraction Module based on semantic similarity. We use averaged word2vec word embeddings [27] to map the names to a vector (embedding) space where cosine similarity is used to match the names. An embedding is a vector representation capturing the syntactic and semantic properties of the word. Distributional hypothesis [14] states that words occurring in similar context have similar meanings. Typically, these embeddings are learned from co-occurrence statistics of words in large text corpora.

Table 1: Linguistic Rules used in the system. Bold words are the specified relation. solid underline is the verb, dashed is the subject and dotted is the main verb's object.

Relation Name	Example Sentence	Extracted State
Prepositional Modifier and its object	Max can go to different places such **as restaurants** and **parks**	(Max, go, restaurant) (Max, go, park)
Prepositional Complement and its object	Max can engage in different activities including **riding** a **horse**.	(Max, engage_in, ride_horse)
Adjectival Complement and its object	Max can be **aware** of his **surroundings**.	(Max, be_aware, surrounding)
Preposition and its object	Max can stand **at** the **bus station**.	(Max, stand, station)
Open Clausal Complement and its object	Max would like to **drink** some juice	(Max, drink, juice)

Consequently, semantically similar words have embeddings which are closer to each other in the vector space [28].

To be robust against noise, generic verbs (e.g. light verbs such as 'be') are filtered out before mapping to the embedding space, if they do not have a context-specific meaning. Finally, each extracted post-condition is checked for probabilistic connotation, by searching for keywords corresponding to a notion of uncertainty, such as "probably, possibly, definitely". Each keyword is assigned a different predefined probability, which quantifies the non-deterministic nature of the post-condition.

4.3 State-Affect Extraction

State-mood rules $\mathcal{R}(\mathcal{S},\mathcal{B})$, state-emotion rules $\mathcal{R}(\mathcal{S},\mathcal{E})$, and state-motivation rules $\mathcal{R}(\mathcal{S},\mathcal{M})$ are extracted in a similar manner as the Affordance Extraction Module with a few minor differences. In the first step, pattern matching is used but with a different set of patterns since the natural sentences in which authors describe states of affect are commonly different than affordance descriptions. For instance, in the sentence "Max will get extremely angry *whenever* he fails his exams.", the part before *whenever* is describing the affective change while the other part describes the state (LHS of the rule). The next step of state extraction and matching is the same for the LHS. For the RHS, the direction and magnitude of the change is obtained by looking for certain adverbs. For instance, adverbs such as 'extremely' or 'very much' represent a high degree of change while others such as 'moderately' stand for moderate degree of change.

4.4 Author Feedback

Sometimes the authors might forget to specify the interactions between states of the agent and its states of affect, or forget to specify the pre/post-conditions of an affordance. The author feedback module attempts to identify such cases. Subsequently, the module comes up with suggestions for the authors.

The module considers the set of all the possible missing rules which could be constructed. In order to show only the relevant suggestions to the author, a subset of the rules are selected. This is done by calculating the similarity between LHS and RHS of the rule and if the score is above a predefined threshold, it is added to the subset of the rules to be presented. The intuition being that if the LHS and RHS of the rule are similar enough, the rule is probably worth reviewing. For example, the state "eating" and the emotion "hunger" are related, which may be described by a rule missed by the author.

For the affordances with few pre/post-conditions, the module asks for additional clarification information. One challenge here is to set the similarity threshold in such a way that it does not filter out too many states while also not suggesting too many irrelevant rules either. For our system, the threshold was determined empirically by experimenting with the system.

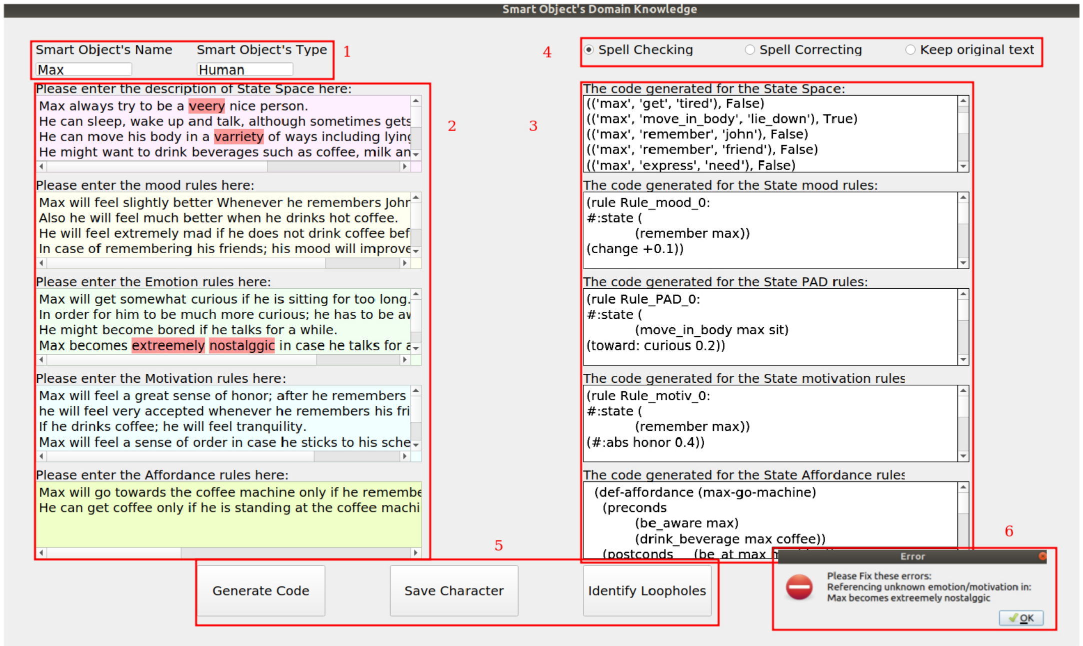
4.5 Common Sense Knowledge Suggestion

ConceptNet [41] is a crowd-sourced online knowledge base of common sense knowledge containing over 8 million entities and 21 million relations. It perfectly suits our purpose since it contains the types of relations that can be used for defining affordances and states. This is in contrast to other knowledge bases like Yago [42] which contain only factual information about the world. The entries and relations in ConceptNet are used in three main ways to assist the authors: (1) For each of the smart-object, the authors also specify a type. These types are used to query the information from ConceptNet relevant to the edge 'IsCapableOf', thereby suggesting to the author possible capabilities or states that are tied to this smart-object; (2) relations such as 'Causes', 'Entails', 'HasFirstSubevent', 'HasLastSubevent, 'CreatedBy', 'HasPrerequisite' are used to suggest pre/post-conditions for the affordances by querying the affordance name; (3) 'CausesDesire' and 'MotivatedByGoal' are used to suggest states which can cause an emotional/motivational response from the agent. These are only suggested if the similarity is above a certain threshold (determined empirically).

Since ConceptNet is crowdsourced, some entries are noisy, for this reason we only choose the more credible entries based on the number of contributors. This is controlled with the WEIGHT parameter. We set it to be at least 1. Table 2 provides some examples of suggestions provided by the proposed Common Sense Knowledge Module. The author is free to accept these suggestions to automatically expand their domain knowledge, or disregard them. As shown in Table 2, suggestions like "Max is a type of Dog, can it learn to do tricks?" can give recommendations to the authors regarding new ideas about aspects of the character previously not investigated.

5 EVALUATION

To help the creative authors use our system as easily and seamlessly as possible, we have developed a simple GUI tool for our application, as shown in Figure 6. This GUI is developed using PyQt5 [43].



Smart Object's Name: Max

Smart Object's Type: Human

● Spell Checking ○ Spell Correcting ○ Keep original text

Please enter the description of State Space here:

Max always try to be a veery nice person.
He can sleep, wake up and talk, although sometimes gets
He can move his body in a varriety of ways including lying
He might want to drink beverages such as coffee, milk an

Please enter the mood rules here:

Max will feel slightly better Whenever he remembers Johr
Also he will feel much better when he drinks hot coffee.
He will feel extremely mad if he does not drink coffee bef
In case of remembering his friends; his mood will improve

Please enter the Emotion rules here:

Max will get somewhat curious if he is sitting for too long.
In order for him to be much more curious; he has to be av
He might become bored if he talks for a while.
Max becomes extreemely nostalggic in case he talks for a

Please enter the Motivation rules here:

Max will feel a great sense of honor; after he remembers
he will feel very accepted whenever he remembers his fri
If he drinks coffee; he will feel tranquility.
Max will feel a sense of order in case he sticks to his sche

Please enter the Affordance rules here:

Max will go towards the coffee machine only if he remembe
He can get coffee only if he is standing at the coffee machi

The code generated for the State Space:

```
(('max', 'get', 'tired'), False)
(('max', 'move_in_body', 'lie_down'), True)
(('max', 'remember', 'john'), False)
(('max', 'remember', 'friend'), False)
(('max', 'express', 'need'), False)
```

The code generated for the State mood rules:

```
(rule Rule_mood_0:
#:state (
        (remember max))
(change +0.1))
```

The code generated for the State PAD rules:

```
(rule Rule_PAD_0:
#:state (
        (move_in_body max sit)
(toward: curious 0.2))
```

The code generated for the State motivation rules

```
(rule Rule_motiv_0:
#:state (
        (remember max))
(#:abs honor 0.4))
```

The code generated for the State Affordance rules

```
 (def-affordance (max-go-machine)
  (preconds
        (be_aware max)
        (drink_beverage max coffee))
  (postconds    (be_at max
```

Please Fix these errors:
Referencing unknown emotion/motivation in:
Max becomes extreemely nostalggic

Figure 6: GUI tool for the system: (1) Specification of the character's name and type. (2) Input panels for natural language description of states and rules. (3) Generated output code from natural language description in (2). (4) Toggle for turning spell checking and correction on or off (see highlighted words in the input panels). (5) Various control buttons during authoring, and (6) pop-up of potential errors.

Table 2: Example suggestions provided to the author by the Common Sense Knowledge Module.

Since "Rio" is a "Bird", can it "prepare nest"?
Since "Max" is a type of "Dog", does it "guide a blind person"?
Is "fatten", a post-condition of "feed"?
Is "have guitar in hands", a cause of "play guitar"?
Since "Max" is a type of "Dog", can it "learn to do tricks" ?

Spacy 2.0 [17] is used for co-reference resolution and dependency parsing due to its good performance, robustness and ease of use. The co-reference resolver was further improved to resolve possessive pronouns as well. Pre-trained word embedding (word2vec) are obtained using Gensim [35], and the PyEnchant library[1] is used for spelling mistake detection and correction.

The task of character development for intelligent virtual agents is a rather new field of study and to our knowledge, there is no standard benchmark or baseline for comparative evaluation of systems. Some authoring tools such as STAR[5] are intended for story comprehension but they have an entirely different objective of identifying the main plots of the story, while in our case the goal is to extract the personality traits of an agent. It is also not clear what should be the most appropriate evaluation metrics. Furthermore bringing the actual character to life needs running a planning mechanism which might not be available in many cases.

Nevertheless, as described in the next sections, for evaluating our tool, we quantitatively compare the output of our system with a "gold standard dataset". We also evaluate our system qualitatively by conducting a user study to assess the system's applicability.

5.1 Quantitative Evaluation

To quantitatively evaluate our system, we developed a "gold standard dataset" to be used for comparison.The dataset was developed by a team of professional writers with diverse writing styles. Manual annotation of the natural language sentences was done by a team of domain experts to identify the corresponding domain knowledge. The dataset contains 26 states, 23 affordances and 6 state affect rules. Each affordance on average has $1 - 2$ pre/post-conditions. By feeding the natural language component of the dataset into the model we generate the corresponding knowledge, which is then compared to the ground truth knowledge.

Results. The tool identified all of the intended states, however it identified 10 extra states as well. Of the 80 affordance pre/post-conditions in the gold standard, 69 were correctly identified, with an accuracy of 86.25%. Out of the 11 cases that did not match, no condition was identified in one case and in the remaining 10 cases, a false condition was identified. Some conditions were especially prone to errors. For example, out of 10 mismatches, "(Max, has, money)" is responsible for 3 mismatches and "(Max, focus, typing)"

[1] https://pypi.org/project/pyenchant/

Table 3: User study results for (a) Tool-specific questionnaire, and (b) System Usability Study (SUS) questionnaire.

	Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Questions about tool	I the tool useful for assisting in the authoring experience.	0.0%	4.8%	33.5%	**47.6%**	14.3%
	It was easy to learn the tool.	9.5%	**33.3%**	23.8%	28.6%	4.8%
	It was easy to use the tool.	0%	23.8%	**33.3%**	28.6%	14.3%
	I would likely use the tool to aid authoring again.	4.8%	9.5%	14.3%	**47.6%**	23.8%
	The suggestions were useful for the authoring experience.	14.3%	19%	23.8%	**28.6%**	14.3%
	The suggestions improved my authoring experience.	23.8%	4.8%	**33.3%**	28.6%	9.5%
	How did you find the quality of the suggestions?	14.3%	9.5%	28.6%	**42.9%**	4.8%
	I think the suggestions were helpful in the creative process.	4.8%	4.8%	38.1%	**42.9%**	9.5%
System Usability Study	I think that I would like to use this system frequently.	9.5%	19.0%	23.8%	**28.6%**	19.0%
	I found the system unnecessary complex	23.8%	**47.6%**	9.5%	14.3%	4.8%
	I thought the system was easy to use	9.5%	19.0%	**28.6%**	**28.6%**	14.3%
	I would need the support of a technical person to be able to use this system.	19.0%	19.0%	**23.7%**	14.3%	**23.9%**
	I found the various functions in this system were well integrated.	0.0%	9.5%	**42.9%**	**42.9%**	4.8%
	I thought there was too much inconsistency in this system.	14.3%	**52.4%**	28.6%	4.8%	0.0%
	I would imagine that most people would learn to use this system very quickly.	14.3%	4.8%	23.8%	**33.3%**	23.8%
	I found the system very cumbersome to use.	14.3%	**33.3%**	**33.3%**	19.0%	0.0%
	I felt very confident using the system.	9.5%	23.8%	19.0%	**42.9%**	4.8%
	I needed to learn a lot of things before I could get going with this system.	14.3%	**33.3%**	9.5%	28.6%	14.3%

is responsible for 4 mismatches. This occurs as in these cases word embeddings are not able to capture the semantics of the state. Also some of the states have a higher number of neighboring states in the embedding space. For example the state "(Max, focus, typing)" was confused 2 times with "(Max, Focus, help_customers)" and also 2 times with "(Max, focus, play)". This illustrates the challenges associated with using state embeddings, with confused states having significant semantic resemblance, e.g., both are about "focusing". In future, we plan to address these issues by exploring other representations for states. Additionally, the system sometimes had difficulty handing complex sentences containing multiple verbs, where the sentence simplification module may have fed incorrect sentences to the state extraction module.

5.2 User Study

We recruited 21 users (11 male, 10 female), between 21 and 49 years of age to evaluate our tool. The subjects were given a short tutorial describing the functions of the tool and asked to author their own character domain. After completing the task, they were asked two sets of questions: (1) tool-specific questions, (2) System Usability Study (SUS) questionnaire [3]. Responses were using a 5 point Likert scale [22] (Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree). The SUS score was 61.2 which is slightly below the average value of 68. The questions and user responses are reported in Table 3. 81% of the users found the GUI of the tool to be convenient and easy to use.

6 CONCLUSION

This paper presents a platform that allows users to author planning domains for intelligent virtual agents using a simple, intuitive natural language interface. We additionally explored the potential for using common sense knowledge to further inspire the creativity of the authors by providing suggestions about the domain. We quantitatively evaluate our tool by evaluating its output on a gold standard dataset. Two user studies are also conducted which highlight the usability and value of the natural language interface. Our platform is currently integrated with a specific agent architecture. However, it can be generalized to universal planning domain definition languages (e.g., PDDL) in a straightforward manner. The ideas can even be further extended to non affordance based social interactive agent architectures as long as these have some consistent representation for their inner workings. The natural understanding component of the proposed system is not perfect and it may face problems in the cases where an input sentence is very long and complex or it describes abstract concept which may be difficult to map to a specific rule. We plan to address these issues in the future.

In our approach sentences were processed individually without taking into account the discourse information that links the states implied in the text. In the future, we would like to consider leveraging discourse information by considering the sequence of states/actions which are described in text. There has been work in the area of modeling event chains and scripts [29–31] and we would like to explore this line of research.

ACKNOWLEDGMENTS

Mubbasir Kapadia has been funded in part by NSF IIS-1703883 and NSF S&AS-1723869.

REFERENCES

[1] Rama Bindiganavale, William Schuler, Jan M Allbeck, Norman I Badler, Aravind K Joshi, and Martha Palmer. 2000. Dynamically Altering Agent Behaviors Using Natural Language Instructions. In *Departmental Papers (CIS)*.

[2] SRK Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning High-Level Planning from Text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[3] John Brooke et al. 1996. SUS-A Quick and Dirty Usability Scale. *Usability Evaluation in Industry* (1996).

[4] Justine Cassell, Joseph Sullivan, Elizabeth Churchill, and Scott Prevost. 2000. *Embodied Conversational Agents*. MIT press.

[5] Irene-Anna Diakidoy, Antonis Kakas, Loizos Michael, and Rob Miller. 2015. STAR: A System of Argumentation for Story Comprehension and Beyond. In *AAAI Spring Symposium Series*.

[6] Matthew Flatt and PLT. 2010. *Reference: Racket*. Technical Report PLT-TR-2010-1. PLT Design Inc.

[7] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. 2003. A Survey of Socially Interactive Robots. *Robotics and Autonomous Systems* (2003).

[8] Patrick Gebhard. 2005. ALMA: a Layered Model of Affect. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[9] Guglielmo Gemignani, Emanuele Bastianelli, and Daniele Nardi. 2015. Teaching Robots Parametrized Executable Plans through Spoken Interaction. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[10] James J Gibson. 2014. *The Ecological Approach to Visual Perception: Classic Edition*. Psychology Press.

[11] Dan Goldwasser and Dan Roth. 2014. Learning from Natural Instructions. *Machine learning* (2014).

[12] Jonathan Gratch and Stacy Marsella. 2004. A Domain-Independent Framework for Modeling Emotion. *Cognitive Systems Research* (2004).

[13] Brian Harrington and Michael O‘Connell. 2016. Video Games as Virtual Teachers: Prosocial Video Game Use by Children and Adolescents from Different Socioeconomic Groups is Associated with Increased Empathy and Prosocial Behaviour. *Computers in Human Behavior* (2016).

[14] Zellig S Harris. 1954. *Distributional Structure*. Englewood Cliffs, NJ: Prentice-Hall.

[15] Thomas Hayton, Julie Porteous, Joao F Ferreira, Alan Lindsay, and Jonathon Read. 2017. StoryFramer: From Input Stories to Output Planning Models.

[16] Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[17] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. (2017).

[18] Marcelo Kallmann and Daniel Thalmann. 1999. Modeling Objects for Interaction Tasks. In *Computer Animation & Simulation*.

[19] Aishwarya Kamath and Rajarshi Das. 2018. A Survey on Semantic Parsing. *CoRR* abs/1812.00978 (2018).

[20] ChanMin Kim and Amy L Baylor. 2008. A Virtual Change Agent: Motivating Preservice Teachers to Integrate Technology in their Future Classrooms. *Journal of Educational Technology & Society* (2008).

[21] Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso. 2013. Learning Environmental Knowledge from Task-Based Human-Robot Dialog. In *IEEE International Conference on Robotics and Automation (ICRA)*.

[22] Rensis Likert. 1932. A Technique for the Measurement of Attitudes. *Archives of psychology* (1932).

[23] Alan Lindsay, Jonathon Read, Joao F Ferreira, Thomas Hayton, Julie Porteous, and Peter J Gregory. 2017. Framer: Planning Models From Natural Language Action Descriptions. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

[24] Yoichi Matsuyama, Arjun Bhardwaj, Ran Zhao, Oscar Romeo, Sushma Akoju, and Justine Cassell. 2016. Socially-Aware Animated Intelligent Personal Assistant Agent.

[25] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL - The Planning Domain Definition Language*. Technical Report.

[26] Albert Mehrabian. 1996. Pleasure-Arousal-Dominance: a General Framework for Describing and Measuring Individual Differences in Temperament. *Current Psychology* (1996).

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Annual Conference on Neural Information Processing Systems (NIPS)*.

[29] Ashutosh Modi. 2016. Event Embeddings for Semantic Script Modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

[30] Ashutosh Modi and Ivan Titov. 2014. Inducing Neural Models of Script Knowledge. In *Conference on Computational Natural Language Learning (CoNLL)*.

[31] Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modelling Semantic Expectation: Using Script Knowledge for Referent Prediction. *Transactions of the Association for Computational Linguistics* 5 (2017), 31–44. https://transacl.org/ojs/index.php/tacl/article/view/968

[32] Florian Pecune, Magalie Ochs, Stacy Marsella, and Catherine Pelachaud. 2016. Socrates: from Social Relation to Attitude Expressions. In *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*.

[33] Vittorio Perera and Manuela Veloso. 2014. Task Based Dialog for Service Mobile Robot. In *AAAI Fall Symposium Series*.

[34] Mihai Pomarlan and John Bateman. 2018. Robot Program Construction via Grounded Natural Language Semantics & Simulation. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

[35] Radim Rehurek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC Workshop on New Challenges for NLP Frameworks*.

[36] Steven Reiss and Susan M Havercamp. 1998. Toward a Comprehensive Assessment of Fundamental Motivation: Factor Structure of the Reiss Profiles. *Psychological Assessment* (1998).

[37] Tiago Ribeiro, André Pereira, Eugenio Di Tullio, Patrıcia Alves-Oliveira, and Ana Paiva. 2014. From Thalamus to Skene: High-level Behaviour Planning and Managing for Mixed-Reality Characters. In *Proceedings of the IVA 2014 Workshop on Architectures and Standards for IVAs*.

[38] Horacio Saggion. 2017. Automatic Text Simplification. *Synthesis Lectures on Human Language Technologies* (2017).

[39] Rushit Sanghrajka, Sasha Schriber, Markus H Gross, and Mubbasir Kapadia. 2018. Computer-Assisted Authoring for Natural Language Story Scripts. In *AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI)*.

[40] Avirup Sil and Alexander Yates. 2011. Extracting STRIPS Representations of Actions and Events. In *International Conference Recent Advances in Natural Language Processing (RANLP)*.

[41] Robert Speer and Catherine Havasi. 2012. Representing General Relational Knowledge in ConceptNet 5.

[42] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a Core of Semantic Knowledge. In *International Conference on World Wide Web*.

[43] Mark Summerfield. 2007. *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming (paperback)*. Pearson Education.

[44] Mariët Theune, Sander Faas, Anton Nijholt, and Dirk Heylen. [n. d.]. The Virtual Storyteller: Story Creation by Intelligent Agents. In *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment Conference (TIDSE)*.

[45] Kristina Yordanova. 2017. TextToHBM: A Generalised Approach to Learning Models of Human Behaviour for Activity Recognition from Textual Instructions. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

[46] Atef Ben Youssef, Mathieu Chollet, Hazaël Jones, Nicolas Sabouret, Catherine Pelachaud, and Magalie Ochs. 2015. Towards a Socially Adaptive Virtual Agent. In *International Conference on Intelligent Virtual Agents (IVA)*.