Getting it Right the Fourth Time: Goal-driven Behavior Using Vector Space Models

1st Nancy Fulda
Department of Computer Science
Brigham Young University
Provo, USA
nfulda@cs.byu.edu

2nd Benjamin Murdoch
Department of Computer Science
Brigham Young University
Provo, USA
ben.murdoch.94@gmail.com

3rd Daniel Ricks
Department of Computer Science
Brigham Young University
Provo, USA
daniel_ricks@byu.edu

***Abstract*—This paper presents a method for imbuing agents with high-level domain knowledge via short command phrases given by a human designer. These command phrases, or goals, are provided at design time and describe the behaviors a human deems necessary in order to succeed at the current task. At run time, each goal is converted into a vector representation and is used to guide the agent's exploration in a text-based environment. At each time step, the agent selects behaviors which it believes are relevant to its current objective, thus reducing the search space and increasing obtained rewards. We show that task-specific goals improve performance in a majority of virtual worlds, and discuss the potential for more general applications.**

***Index Terms*—commonsense reasoning, knowledge representation, human-computer interaction, vector space models**

I. Introduction

Reinforcement learning is a laborious process, often requiring tens of thousands of failed attempts before a desirable result is obtained. This is acceptable in simulated environments, but becomes prohibitive when working with humans or in domains with time and energy constraints. Ideally, we would like to create agents that are able to learn the way humans do, requiring only a few attempts before discovering a beneficial behavior. We want agents that can 'get it right' on the first try; or at the very least, on the fourth, fifth or sixth try.

Inspired by the human ability to transfer domain knowledge via spoken language, we present a method whereby agents navigating a text-based world are imbued with high-level goals defined by a human user. The agent evaluates these free-form instructions using a vector space model that applies commonsense knowledge in the form of canonical analogy vectors. The extracted knowledge allows the agent to infer (a) which of the available objects are relevant to its current goal, and (b) which possible behaviors are simultaneously relevant to the goal and reasonable to attempt in conjunction with the given object. A key advantage of this approach is the general ease of use. No specialized knowledge is required to formulate instructions for the agent. One simply specifies two-word commands like 'find books' or 'create light'.

We apply our method in a setting intended to mimic real-world constraints. Rather than allowing the agent to try as many actions as necessary until it finds the right behavior, we require the agent to select a small subset of its available action space. This forces the agent to behave in a more focused manner, and brings its chosen actions more directly in alignment with its goals. We are encouraged to note that the benefits of our goal-directed method become more pronounced as the number of allowed actions decreases. In other words, the more closely we align our simulation with reality, the more valuable human input becomes for the agent.

Our primary research intent is to take a step in the direction of autonomous agents with resilient cognitive abilities. Long-term, we desire agents that can analyze their environments, select goals in a self-directed manner, and act upon those goals in focused and reasonable ways. This paper addresses the latter-most objective: acting on goals in focused and reasonable ways.

II. Related Work

Vector space models, in which words or groups of words are represented as n-dimensional vectors, have been an active area of research since the 1960s [14]. In recent years, statistical or count-based models have been replaced by embeddings trained from large amounts of uncurated data, although some researchers suggest that the type of embedding algorithm used may matter less than the hyperparameters used during the creation process [26] [11] [18].

In our work, we utilize the skip-thought embedding model presented by Kiros et al. in 2015 [13], which in turn relies on the word2vec embeddings trained by Mikolov et al. [19]. Word embeddings created using algorithms such as word2vec, skip-gram [20], Glove [23], Latent Dirichlet Allocation [1], and simple co-occurrence matrices [3] can be used for analogical reasoning tasks such as identifying similarities between words or associating cities with their capitals [21]. They have also been applied to more complex reasoning tasks [9] [2].

Skip-thought vectors [13] are an application of the word2vec skip-gram training method at the sentence level, resulting in an encoder that represents each input sentence as a fixed-length vector. (Similar work is also found in the paragraph vectors of Le and Mikolov [17], Google's Universal Sentence Encoder [4], and the InferSent embedding model [5].) Our work builds on the semantic structure of the skip-thought embedding space and on the affordance-based reasoning methods of [8] in order to create an agent capable of accepting user-defined goals and

GOAL	VERB	NOUN
unlock door	turn	key
carry water	fill	bottle
wash floor	use	sponge
catch butterfly	swing	net
reach roof	climb	staircase
study geometry	read	textbook
enter castle	cross	drawbridge
enter spaceship	use	airlock
take bath	fill	bathtub
activate machine	flip	switch

Fig. 1. Sample canonical analogy set used to find goal-driven actions. The full set used in our experiments included 17 goal-verb pairs and 18 goal-object pairs. Notably, a number of critical in-game objects (such as 'door', 'window', 'candy' and 'lantern') do not appear anywhere in the canonical examples.

behaving in a manner conducive to achieving them. In some respects, this research resembles the work of Kaplan, Sauer and Sosa in natural language guided gaming [12], in that both architectures enable the agent to improve its performance as a result of text input from a human user. The research differs, however, in the way language understanding is acquired and applied. [12] learn visual and language embeddings together in order to play a visual game. Our agent performs no learning beyond the pre-trained vector space model. Instead, it uses the model's embeddings as a common-sense knowledge base in order to find concrete actions that support an abstractly-state goal.

Our agent architecture resembles the work of [8], [15] and [22], in which an agent interacts repeatedly with a text-based game engine. Test environments were obtained from the Autoplay learning environment for interactive fiction [25] and results are compared to those reported by previous researchers. These text-based virtual worlds are a challenging and largely unsolved domain: with over 85,000 nouns and at least 10,000 verbs in the English language, a simple verb/noun action phrase (such as 'climb tree' or 'eat apple') can be composed in over 8.5 x 10^8 distinct ways, most of which are utter nonsense. The significance and complexity of these virtual worlds has been explored by [16], who identified them as a step toward general problem solving.

III. Goal-Directed Action Selection

When attempting to master new tasks, autonomous agents frequently resort to trial and error. The agent is not familiar with anything, and so it attempts *everything*. We seek to break away from this pattern by imbuing our agents with human-generated domain knowledge. If a human operator could share her understanding of the environment with the agent, then the exploration and learning process would be greatly accelerated.

With this goal in mind, we provide our agent with a set of pre-defined instructions specific to each task. These instructions are abstract and generalized, like 'open things' or 'get stuff'. From this abstract, human-defined domain knowledge, the agent seeks to identify concrete behaviors which will lead to the fulfillment of its goals.

We model our agent as a purely text-based entity. The agent's state s is a natural language string describing the immediate environment. Actions are represented as verb/noun pairs $a = v + \text{`\ '} + n$ where v is an English-language verb and n is an English-language noun that has been extracted from s using standard part-of-speech tagging. The agent's objective is to obtain game points by interacting with the environment. Different environments award points for different behaviors, but all environments require a minimum level of 'reasonableness' in the agent's actions, and will not award points for nonsense behaviors.

In order to ensure proper state space disambiguation, the agent executes a 'look' command on every second iteration and interprets the resulting game text as its current state. This limits the agent's ability to obtain points, but creates a better testing environment for the behaviors we are studying. For similar reasons, the agent was not given information about items carried by the player and was also not given the ability to execute prepositional phrases such as 'put book on shelf'. Because we are interested primarily in the agent's ability to efficiently navigate its action space, we also do not implement a learning model. State-space obfuscation caused by score reporting is prevented by stripping all numerals from the game text.

Algorithm 1 Our Agent: Initialization and Control Loop

```
n = final number of nouns desired
v = final number of verbs desired
goal_list = a set of user-defined goals, represented as text
navigation_verbs = ['north', 'south', 'east', 'west', 'northeast', 'southeast',
    'southwest', 'northwest', 'up', 'down', 'enter']
manipulation_verbs = ['get', 'drop', 'push', 'pull', 'open', 'close', 'search',
    'attack']
for 0 to 2000 do
    state = game response to last action
    goal = random.choice(goal_list)
    goal_vector = skip-thought encoding of goal text
    top_nouns = results of Algorithm 2
    top_verbs = results of Algorithm 3
    noun = randomly selected noun from the top_nouns
    verb = randomly selected verb from the top_verbs
    execute action verb + ' ' + noun
end for
```

Given a specific goal g, our agent seeks to identify action tuples (v_i,n_i) such that the natural language statement $v+$ ' ' $+n$ describes a behavior that is supportive of the goal. This is done within the skip-thought vector space model.

Let $goal$ be a natural language goal and v_goal be the vector encoding of $goal$. Let (g_1,v_1) ... (g_i,v_i) be a set of encoded canonical examples mapping sample goals to verbs which help facilitate those goals (see Figure 1, first and second columns) and let (g_1,n_1) ... (g_i,n_i) be a set of encoded canonical examples mapping sample goals to objects which help facilitate those goals (see Table X, columns 1 and 3).

We define a canonical goal-to-verb vector V_{verb} = $1/k$ $\sum_0^k$ (v_i - g_i). The canonical goal-to-noun vector is defined similarly: V_{noun} = $1/k$ $\sum_0^k$ (n_i - g_i).

GAME	HUMAN-DEFINED GOALS
zork1	‘enter house’, ‘get stuff’, ‘create light’, ‘move furniture’, ‘climb tree’, ‘unlock locks’
zork2	‘enter buildings’, ‘get stuff’, ‘create light’, ‘move things’
zork3	‘enter buildings’, ‘get stuff’, ‘create light’, ‘move things’
candy	‘get candy’, ‘search for candy’
omniquest	‘get stuff’, ‘climb tree’, ‘wear clothing’, ‘move things’, ‘dig’
bunny	‘enter holes’, ‘get stuff’, ‘move things’, ‘open things’, ‘burn monsters’, ‘unlock locks’
detective	‘get stuff’, ‘enter buildings’
mansion	‘unlock locks’, ‘take stuff’, ‘turn on’, ‘turn off’
spirit	‘open things’, ‘get scrolls’
zenon	‘get stuff’, ‘look under bed’, ‘unlock locks’, ‘turn off light’
cavetrip	‘open furniture’, ‘search furniture’, ‘get clothes’, ‘get food’, ‘get batteries’
parc	‘enter buildings’, ‘get stuff’, ‘close curtains’

Fig. 2. The human-defined goals used in each game. The goals were formulated based on scoring within the specific game.

Algorithm 2 Noun Prioritization

Require: $goal\_vector$
Ensure: a list of nouns
$noun\_analogy\_vector$ = $goal\_vector$ + canonical goal-to-noun vector (see Section x.y)
$game\_text$ = input text from the game engine
$nouns$ = nouns extracted from $game\_text$ using NLTK
$distances = []$
for all $noun$ in $nouns$ **do**
    $noun\_vector$ = skip-thought encoding of $noun$
    $distances$.append(cosine_distance(noun_vector, noun_analogy_vector))
end for
top_nouns = take n nouns, ordered by associated distances (closest first)
return top_nouns

Algorithm 3 Verb Prioritization

Require: $goal\_vector$, top_nouns
Ensure: a list of verbs
$verb\_analogy\_vector$ = $goal\_vector$ + canonical goal-to-verb vector (see Section x.y)
$distances = []$
$noun$ = randomly selected noun from top_nouns + the empty noun ‘’
if $noun$ == ‘’ **then**
    execute a random selection from $navigation\_verbs$
end if
$affordance\_verbs$ = the 60 most affordant verbs for $noun$
for all $verb$ in $affordance\_verbs$ + $manipulation\_verbs$ **do**
    v_verb = skip-thought encoding of $verb$
    $distances$.append(cosine_distance(verb_vector, verb_analogy_vector))
end for
top_verbs = take n verbs, ordered by associated distances (closest first)
return top_verbs

On each time step, our agent extracts available nouns from the game text and prioritizes them based on increasing cosine distance from the point v_goal + V_{noun}. This list is then truncated according to a user-defined constraint on the number of nouns the agent is allowed to evaluate. A noun is selected at random from the truncated list, and the agent proceeds to seek a verb that is simultaneously affordant to the selected noun [10] and relevant to the current goal. Candidate verbs are selected by using the affordance vector V_{aff} as described in [8]. These candidates are prioritized by decreasing proximity to v_goal + V_{noun}, and again the list is truncated based on user-defined constraints. A goal is selected at random from the truncated set and the action $v + \text{‘ ’} + n$ is executed.

IV. Results

A. Experimental Setup

To evaluate our methods, we selected 12 games from the autoplay repository that exhibited sufficient structural and narrative cohesion for our purposes.

On each time step, our agent randomly selected a goal and attempted to find a goal-directed action as described in Section 3. We compared its performance with two competitors: (1) A baseline agent that randomly paired nouns and verbs. Nouns were extracted from the game text, but were not filtered or prioritized. Verbs were sampled randomly from the 1000 most commonly appearing verbs in Wikipedia. (2) The second competitor was an affordance agent that tried to pair nouns with appropriate verbs. Nouns were extracted from the game text, but were not filtered or prioritized. Verbs were selected according to the affordance analogy methods presented by [8].

In order for our agent to behave in a goal-directed fashion, it needed a set of task-specific goals for each trial game. These goals were acquired by allowing a human to examine the early stages of each game and provide a set of behaviors he or she felt were most conducive to point acquisition. The goals were passed to the agent as a list of natural-language imperatives, and were subsequently converted to skip-thought vectors. For the most part, these goals were phrased as generic actions which the agent then transposed into specific behaviors. For example, in order to achieve the goal ‘enter house’ in *zork1*, the agent must execute the command ‘open window’ followed by ‘enter window’.

It is important to note that the agent did not use the text of the goals directly. For example, when instructed to ‘burn monsters’ in *bunny*, the agent did not extract the verb ‘burn’ or the noun ‘monsters’ from the goal text. Instead, it encoded the goal as a 4800-dimensional vector and from that was able to correctly identify verb/noun pairs that would lead to points. (In this case, the required command was ‘burn ooze’, and it only works if the player has already executed the command ‘get torch’.)

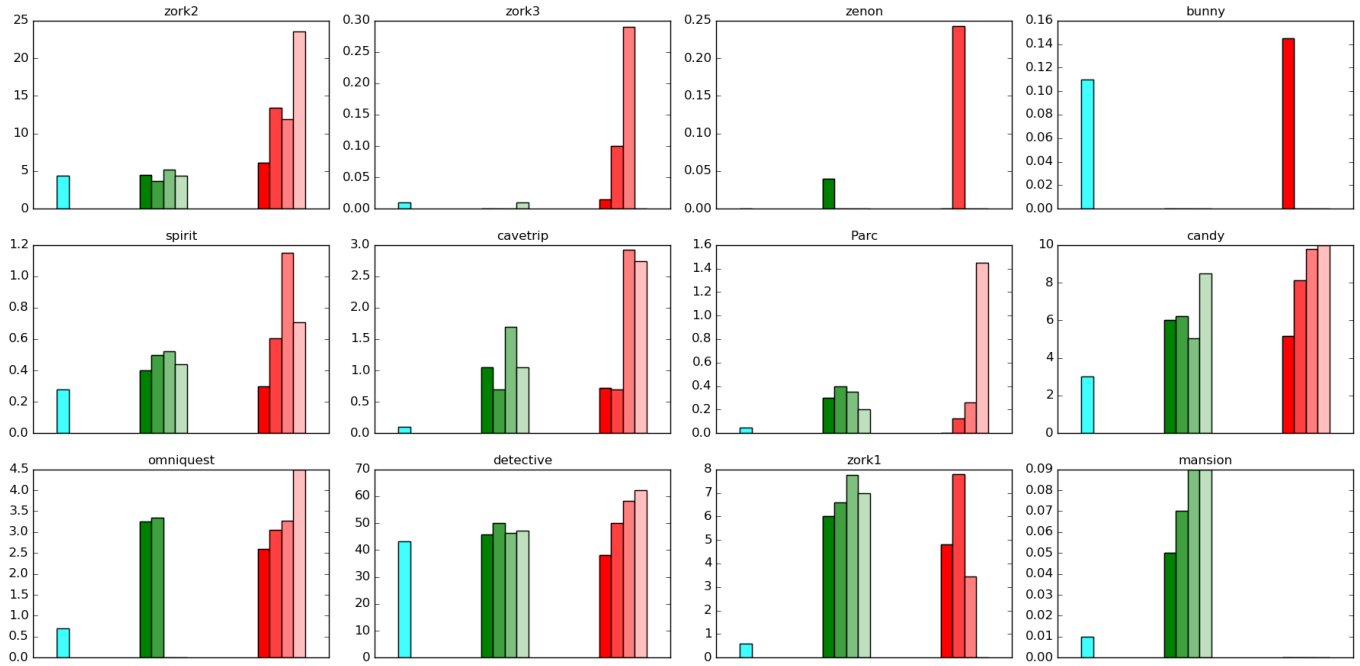

Fig. 3. Agent performance with increasing constraints. Cyan: random actions; green: affordance-based actions; red: goal-directed actions. From left to right for each 4-bar set, the agent was permitted to try (a) 30 verbs (b) 15 verbs (c) 3 verbs (d) 1 verb. The goal-directed agent, in addition to verb constraints, restricted itself to (a) 15 nouns (b) 5 nouns (c) 3 nouns (d) 1 noun. The results from 45 data runs were averaged to produce the figure.

B. Performance Against Baseline Methods

Figure 3 shows the performance of our Goal-Directed Agent against the random and affordant baselines. Because the random agent has no criteria for selecting a subset of verbs from its 1000-verb base set, it is represented in the figure by only a single cyan bar. The affordant agent (green) and the Goal-Directed Agent (red) are each depicted with four bars, one for each of the constraint options in Table 1.

Examination of Figure 3 reveals that in almost every case, the highest game score was achieved by the Goal-Directed Agent. In 8/12 cases, the Goal-Directed Agent's performance steadily improved as constraints increased, indicating that the human guidance had a profound impact on its performance. In five cases, the agent's 1-noun-1-verb implementation achieved the highest score of *all* agent trials: The agent 'got it right' the first time.

C. The Importance of Choosing the Right Goal

Figure 3 immediately raises a critical question: Might it be the reduction in search space size, and not the specific nature of the reduction, that produces the observed performance improvements? We tested this notion by comparing four different variants of the Goal-Directed Agent. The first variant used the task-specific goals shown in Figure 4. The second variant used a generic goal set designed to be generally relevant across all text-based adventure games. The generic goals were: *'enter buildings', 'get stuff', 'move things', 'open things', 'search furniture', 'attack enemies', 'unlock locks'*. The third variant used a set of counterproductive goals designed to thwart the agent's attempts to earn points. The counterproductive goals were: *'close things', 'drop stuff', 'eat penguins', 'murder blueberries'*. The fourth agent variant used no goals at all, and instead reduced the number of verbs tried per noun using a randomized method.

Results are shown in Figure 4. In eleven cases out of twelve, the task-specific goals resulted in superior or identical performance to the other goal schema. In five cases out of twelve, the task-specific goals were clearly superior, and in one case out of the twelve, the task-specific goals produced inferior performance. While there is still clearly room for improvement in these results, we consider them a strong indication that goal-directed action selection (and not merely a reduction in the number of nouns tried) contributes significantly to the agent's performance.

V. General Application

Encouraged by the performance of our algorithm in the specific domain of text-based adventure games, we attempted to apply our method to a more general problem. We conceived of an 'oracle' that accepted a goal from a human user and then, without any additional input, proposed a set of actions conducive to that goal.

When given a relatively small, human-defined set of objects to choose from, the system showed inklings of promise. For example, given the goal 'enter house' and the object set *'lantern', 'house', 'door', 'key', 'table', 'window', 'lantern', 'floor', 'boom', 'mop', 'stove', 'food', 'doorknob', 'sword', 'knife', 'battleaxe', 'soap'*, the agent proposed the following

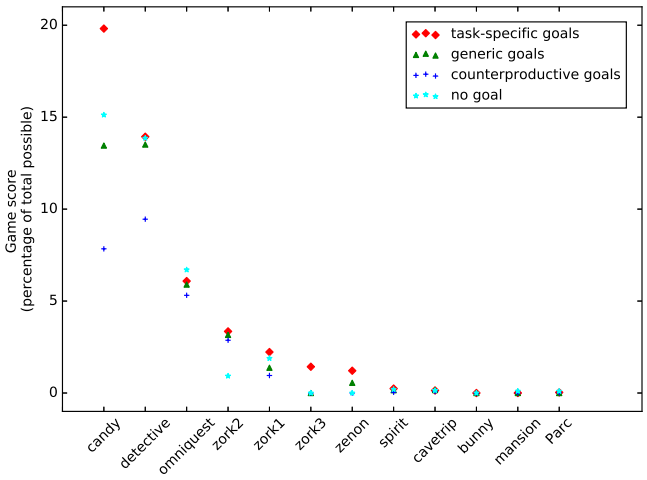

Fig. 4. Agent performance as a function of goal quality. Average of 45 trials of 2000 steps each. Red triangles indicate human-provided goals that were specifically crafted for the problem domain. Green triangles indicate goals that are generally applicable in text-based adventure games, but that are not specifically targeted to the game being played. Blue crosses indicate goals that were deliberately designed to thwart agent performance, and cyan stars indicate performance when the action space was randomly reduced without reference to any particular goal.

actions: *'relocate house', 'reassemble doorknob', 'enter door', 'use key', 'utilize food'*. Given the same object list and the goal to 'create peace' the agent proposed: *'utilize food', 'use key', 'forsake sword', 'eviscerate knife', 'nominate mop'*. Both cases involved several ridiculous combinations, but also had instances of impressive insight (e.g. 'enter door' and 'forsake sword').

When we attempted to make the oracle more general, however, by giving it a set of 1000 nouns to choose from, the results became utterly nonsensical. For 'enter house', the agent proposed *'convene house', 'situate library', 'enter hall', 'climb door', 'unlock room'*. Given the goal 'create peace' it suggested: *'govern peace', 'revert nomination', 'revert consensus', 'transcend genre', 'revert template'*. We conclude that a generalized application of this method will require a vector space model with stronger analogical properties. As several interesting lines of research are exploring this possibility [6], [28], we are hopeful that generalized applications of this method will soon become feasible.

VI. Future Work

Future work in this area should focus first and foremost on the development of vector space models with optimal analogical properties. This could potentially be accomplished by applying transformers [27] or other attention-based neural network architectures to the specific task of learning vector spaces that satisfy prespecified analogical criteria. It might also be possible to apply a lightweight transformation to the hidden activations of advanced language models like transformer-XL [7] and GPT-2 [24] in order to regularize the distinctions between the embedded representations of input texts.

A second area of exploration should include the application of descriptive rather than imperative guidance scripts. When humans teach each other how to achieve a task, they are more likely to say things like 'You're going to need some batteries' than they are to say 'get batteries'. It seems logical that true human-computer interaction will tend to reflect the former rather than the latter behavior pattern, and future research in this area should reflect that.

VII. Conclusion

This paper presents a method for imbuing agents with high-level domain knowledge via human-generated guidance phrases. We have shown that a sentence-level vector space model can be used to prioritize possible actions in a text-based domain. When equipped with goals that accurately reflect the environment, our agent outperforms both random exploration and affordance detection methods. When given faulty goals, agent performance declines. When given task-specific goals and constrained to select only its highest-ranked actions, agent performance improves.

Although this method works well in our experimental domain, we seek to make it more generally applicable. We believe this can be accomplished by identifying or training a vector space model that is uniquely optimized for our research area. Such an embedding space could conceivably contain structural representations of causation and temporal relations as well as more general knowledge.

References

[1] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[2] Georgios-Ioannis Brokos, Prodromos Malakasiotis, and Ion Androutsopoulos. Using centroids of word embeddings and word mover's distance for biomedical document retrieval in question answering. *CoRR*, abs/1608.03905, 2016.

[3] Curt Burgess and Kevin Lund. Modelling parsing constraints with high-dimensional context space. 12, 03 1997.

[4] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[5] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[6] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

[7] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[8] Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1039–1045, 2017.

[9] Nancy Fulda, Nathan Tibbetts, Zachary Brown, and David Wingate. Harvesting common-sense navigational knowledge for robotics from uncurated text corpora. In *Proceedings of the First Conference on Robot Learning (CoRL) - forthcoming*, 2017.

[10] James J. Gibson. The theory of affordances. In Robert Shaw and John Bransford, editors, *Perceiving, Acting, and Knowing*. 1977.

[11] Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. 2016.

[12] Russell Kaplan, Christopher Sauer, and Alexander Sosa. Beating atari with natural language guided reinforcement learning. *CoRR*, abs/1704.05539, 2017.

[13] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[14] Sheldon Klein, Stephen L Lieman, and Gary Lindstrom. Diseminer: a distributional-semantics inference maker. 1966.

[15] Bartosz Kostka, Jaroslaw Kwiecien, Jakub Kowalski, and Pawel Rychlikowski. Text-based adventures of the golovin AI agent. *CoRR*, abs/1705.05637, 2017.

[16] John E. Laird and Michael van Lent. Human-level AI's killer application: Interactive computer games. *AI Magazine*, 22(2):15–26, 2001.

[17] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[18] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, pages 3111–3119. Curran Associates, Inc., 2013.

[21] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. Association for Computational Linguistics, May 2013.

[22] Karthik Narasimhan, Tejas D. Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *CoRR*, abs/1506.08941, 2015.

[23] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, 2014.

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[25] Daniel Ricks. Autoplay: a learning environment for interactive fiction. https://github.com/danielricks/autoplay, 2016.

[26] Sebastian Ruder. An overview of word embeddings and their connection to distributional semantic models. http://blog.aylien.com/overview-word-embeddings-history-word2vec-cbow-glove, 2016.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[28] Xunjie Zhu, Tingfeng Li, and Gerard De Melo. Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 632–637, 2018.