



Theses and Dissertations

---

2019-08-01

## Semantically Aligned Sentence-Level Embeddings for Agent Autonomy and Natural Language Understanding

Nancy Ellen Fulda  
*Brigham Young University*

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>

---

### BYU ScholarsArchive Citation

Fulda, Nancy Ellen, "Semantically Aligned Sentence-Level Embeddings for Agent Autonomy and Natural Language Understanding" (2019). *Theses and Dissertations*. 7550.

<https://scholarsarchive.byu.edu/etd/7550>

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact [scholarsarchive@byu.edu](mailto:scholarsarchive@byu.edu), [ellen\\_amatangelo@byu.edu](mailto:ellen_amatangelo@byu.edu).

Semantically Aligned Sentence-Level Embeddings for Agent  
Autonomy and Natural Language Understanding

Nancy Ellen Fulda

A dissertation submitted to the faculty of  
Brigham Young University  
in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy

David Wingate, Chair  
Kevin Seppi  
Tony Martinez  
Mike Jones

Department of Computer Science  
Brigham Young University

Copyright © 2019 Nancy Ellen Fulda

All Rights Reserved

## ABSTRACT

### Semantically Aligned Sentence-Level Embeddings for Agent Autonomy and Natural Language Understanding

Nancy Ellen Fulda  
Department of Computer Science, BYU  
Doctor of Philosophy

Many applications of neural linguistic models rely on their use as pre-trained features for downstream tasks such as dialog modeling, machine translation, and question answering. This work presents an alternate paradigm: Rather than treating linguistic embeddings as input features, we treat them as common-sense knowledge repositories that can be queried using simple mathematical operations within the embedding space, without the need for additional training. Because current state-of-the-art embedding models were not optimized for this purpose, this work presents a novel embedding model designed and trained specifically for the purpose of “reasoning in the linguistic domain”.

Our model jointly represents single words, multi-word phrases, and complex sentences in a unified embedding space. To facilitate common-sense reasoning beyond straightforward semantic associations, the embeddings produced by our model exhibit carefully curated properties including *analogical coherence* and *polarity displacement*. In other words, rather than training the model on a smorgasbord of tasks and hoping that the resulting embeddings will serve our purposes, we have instead crafted training tasks and placed constraints on the system that are explicitly designed to induce the properties we seek. The resulting embeddings perform competitively on the SemEval 2013 benchmark and outperform state-of-the-art models on two key semantic discernment tasks introduced in Chapter 8.

The ultimate goal of this research is to empower agents to reason about low-level behaviors in order to fulfill abstract natural language instructions in an autonomous fashion. An agent equipped with an embedding space of sufficient caliber could potentially reason about new situations based on their similarity to past experience, facilitating knowledge transfer and one-shot learning. As our embedding model continues to improve, we hope to see these and other abilities become a reality.

Keywords: Nancy Fulda, dissertation, knowledge representation, linguistic embeddings, word2vec, FastText, universal sentence encoder, BERT embeddings, sentence embedding, common-sense reasoning

## ACKNOWLEDGMENTS

No woman is an island, and I have been pleased to study under the guidance of many worthy scholars. Particular thanks is due to my advisor, David Wingate, whose insights, expectations, and unwavering support are directly responsible for this manuscript. It is rare to find so much generosity and intelligence in one person, and even rarer if he is willing to sprint through a crowded airport to rescue the forgotten laptop of an exhausted grad student. I could not ask for a better mentor.

Additional thanks go to Dan Ventura, who taught me to think analytically and never let me get away with a single typo, and to Todd Peterson, who took on a brashly clueless undergraduate and gave her space to stretch her wings. Thanks to Kevin Seppi for his boundless confidence and insightful critiques, to Tony Martinez for stepping up in a time of need, and to Mike Jones for unflinchingly signing every single dang form twice. Likewise, I am grateful to my lab mates and collaborators Daniel Ricks, Ben Murdoch, Zachary Brown, Nathan Tibbetts, Tyler Etchart and Will Myers. Thanks for the laughs, the puns, and the memories.

Most of all, I would like to thank my husband Fabian, who has been a model of sacrifice, encouragement, and dedication, who knows me better than I know myself, and whose company makes even the most aggravating journeys worthwhile. This dissertation would not have happened without him.

## Table of Contents

<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xviii</b>
<b>I Overview</b>	<b>1</b>
<b>II The Quest for Common-Sense Knowledge</b>	<b>8</b>
<b>1 What Can You Do with a Rock?</b>	
<b>Affordance Extraction via Word Embeddings</b>	<b>10</b>
1.1 Introduction . . . . .	10
1.2 Related Work . . . . .	11
1.3 Wikipedia as a Common Sense Knowledge Base . . . . .	13
1.3.1 Verb/Noun Affordances . . . . .	15
1.3.2 Identifying Graspable Objects . . . . .	17
1.4 Test Environment: A World Made of Words . . . . .	19
1.4.1 Learning Algorithm . . . . .	20
1.5 Results . . . . .	22
1.5.1 Alternate Reduction Methods . . . . .	24
1.5.2 Fixed-length Vocabularies vs. Free-form Learning . . . . .	26
1.6 Conclusion . . . . .	26

<b>2</b>	<b>Harvesting Common-Sense Navigational Knowledge for Robotics from Uncurated Text Corpora</b>	<b>28</b>
2.1	Introduction . . . . .	29
2.2	Related Work . . . . .	30
2.3	Embeddings and Analogies on Hyperspheres . . . . .	31
2.4	Improving Analogies with Canonical Vectors and Directional Scoring . . . . .	33
2.4.1	Canonical analogy vectors . . . . .	34
2.4.2	Scoring Algorithm . . . . .	34
2.5	Quantitative Analysis . . . . .	35
2.5.1	Algorithms tested . . . . .	35
2.5.2	Datasets used . . . . .	37
2.5.3	Results . . . . .	38
2.6	Qualitative Analysis . . . . .	38
2.7	Conclusion . . . . .	41
<b>3</b>	<b>Embedding Grammars</b>	<b>44</b>
3.1	Introduction . . . . .	44
3.2	Background . . . . .	45
3.3	Embedding Grammars . . . . .	46
3.3.1	Subspace region definition . . . . .	48
3.4	A Python Implementation . . . . .	48
3.5	Experiments . . . . .	49
3.5.1	Word extensions . . . . .	49
3.5.2	Intent matching . . . . .	51
3.6	Learning and Extensions to PCFGs . . . . .	51
3.7	Conclusion . . . . .	52
<b>4</b>	<b>Informing Action Primitives Through Free-Form Text</b>	<b>53</b>

4.1	Introduction . . . . .	53
4.2	Related Work . . . . .	55
4.3	Our Algorithm . . . . .	56
4.4	Results . . . . .	59
4.5	Conclusion and Future Work . . . . .	61

### **III Beyond Words and Phrases: Linguistic**

#### **Reasoning at the Sentence Level 62**

##### **5 Threat, Explore, Barter, Puzzle: A Semantically-Informed**

###### **Algorithm for Extracting Interaction Modes 64**

5.1	Introduction and Related Work . . . . .	64
5.2	Methodology . . . . .	66
5.3	Results . . . . .	68
5.4	Conclusions and Future Work . . . . .	75
5.5	Acknowledgements . . . . .	76

##### **6 Conversational Scaffolding: An Analogy-Based Approach**

###### **to Response Prioritization in Open-Domain Dialogs 77**

6.1	Introduction . . . . .	77
6.2	Related Work . . . . .	79
6.2.1	Utterance Retrieval . . . . .	79
6.2.2	The analogical structure of embedding spaces . . . . .	79
6.3	Conversational Scaffolding . . . . .	80
6.3.1	Contextual alignment . . . . .	80
6.3.2	Candidate Response Scoring . . . . .	85
6.4	Experimental Setup . . . . .	87
6.4.1	Text Corpora . . . . .	87

6.4.2	Evaluation Task . . . . .	88
6.4.3	Baselines . . . . .	89
6.5	Results . . . . .	90
6.5.1	Generalization Across Datasets . . . . .	92
6.5.2	Retrieval from Larger Data Repositories . . . . .	92
6.6	Conclusions and Future Work . . . . .	95
<b>7</b>	<b>BYU-EVE: Mixed Initiative Dialog via Structured Knowledge</b>	
	<b>Graph Traversal and Conversational Scaffolding</b>	<b>96</b>
7.1	Overview . . . . .	97
7.1.1	Open Domain Dialog: Challenges and Opportunities . . . . .	97
7.1.2	Invisible Walls vs. Infinite Horizon: How Open Is ‘Open’? . . . . .	97
7.1.3	Text Retrieval vs. Text Generation: Between a Rock and a Hard Place . . . . .	98
7.1.4	Choosing What to Say: A Question of Composition . . . . .	100
7.2	Knowledge Representation . . . . .	101
7.2.1	Property Graph vs. Resource Description Framework (RDF) . . . . .	102
7.2.2	Wikidata . . . . .	103
7.2.3	Knowledge Graph Hosting . . . . .	104
7.3	User and Self Modeling . . . . .	104
7.3.1	Current Implementation of Dynamic Knowledge Graph Models . . . . .	105
7.3.2	A Note on Privacy . . . . .	109
7.4	Response Generation . . . . .	110
7.4.1	Emotive Generators . . . . .	110
7.4.2	Knowledge Graph Generators . . . . .	111
7.4.3	API Generators . . . . .	111
7.5	Response Evaluation . . . . .	112
7.5.1	Conversational Scaffolding: An Analogical Approach to Response Prioritization . . . . .	112



7.5.2	The Chit-Chat Dataset . . . . .	113
7.5.3	Scaffolding Algorithms . . . . .	115
7.6	Supporting Infrastructure . . . . .	121
7.6.1	Topic Detection and Entity/Keyword Extraction . . . . .	121
7.6.2	Offensive Speech Detection . . . . .	121
7.6.3	Intent Recognition . . . . .	122
7.6.4	Response Evaluation (Ranking & Filtering) . . . . .	122
7.6.5	Response Combination . . . . .	126
7.7	Future Work . . . . .	127
7.7.1	Improved Knowledge Graph Structure and Traversal . . . . .	127
7.7.2	Optimized Embeddings Module . . . . .	128
7.7.3	Similarity Metrics for Data Retrieval . . . . .	128
7.7.4	ScriptDog: A Language for Managing Conversational State . . . . .	129
7.8	Conclusion . . . . .	130

## **IV A Well-formulated Embedding Space 131**

<b>8</b>	<b>That is Not What I Said: Improved Discrimination of Negated Sentences via Deep Constrained Neural Embeddings</b>	<b>133</b>
8.1	Introduction . . . . .	133
8.2	Related Work . . . . .	136
8.3	A Deep Constrained Neural Network . . . . .	137
8.4	Network Architecture . . . . .	139
8.5	Semantic Discernment Datasets . . . . .	140
8.6	Quantitative Analysis . . . . .	142
8.7	Qualitative Analysis . . . . .	143
8.8	Conclusion . . . . .	144

V The Road Forward	146
References	148

## List of Figures

1	100-dimensional word vectors projected into a 2-dimensional space demonstrate impressive semantic structure. Common household items are clustered in specific relationships to the rooms in which they tend to be located. The angle of correspondence is at least as significant as proximity. Of particular interest is the word ‘mirror’, which is aligned both with the items found in a bedroom and with the items found in a bathroom. . . . .	2
1.1	Word vectors projected into the space defined by $vector[‘king’] - vector[‘queen’]$ and $vector[‘woman’] - vector[‘man’]$ . In this projection, masculine and feminine terms are linearly separable. . . . .	14
1.2	Verb associations for the noun ‘sword’ using three different methods: (1) Affordance detection using word vectors extracted from Wikipedia, as described in this section, (2) Strict co-occurrence counts using a Wikipedia corpus and a co-occurrence window of 9 words, (3) Results generated using ConceptNet’s CapableOf relationship. . . . .	16
1.3	Word vectors projected into the space defined by $vector[‘forest’] - vector[‘tree’]$ and $vector[‘mountain’] - vector[‘pebble’]$ . Small, manipulable objects appear in the lower-left corner of the graph. Large, abstract, or background objects appear in the upper right. An object’s manipulability can be roughly estimated by measuring its location along either of the defining axes. . . . .	18
1.4	Sample text from the adventure game Zork. Player responses follow a single angle bracket. . . . .	20

1.5	Learning trajectories for eighteen Z-machine games. No agent scored any points on the remaining 31 games. Average of 10 data runs. *In zork3, standard smoothing obscured small, sporadic positive rewards obtained by our reduction algorithms. . . . .	23
1.6	Five verb space reduction techniques compared over 100 exploration epochs. Average of 5 data runs. Results were normalized for each game based on the maximum reward achieved by any agent. . . . .	25
1.7	Sample exploration actions produced by a Q-learner with and without affordance detection. The random agent used nouns extracted from game text and a verb list comprising the 200 most common verbs in Wikipedia. . . . .	26
2.1	Vector operations in a 2-dimensional slice of an n-dimensional unit hypersphere. The three illustrations on the right represent a possible failure mode. In the middle two illustrations, when analogy vector $\vec{AB}$ is applied to point C, the resulting endpoint tends to lie close to one of A, B, or C. The specific failure depends on the angle $\theta$ and the length of $\vec{AB}$ . . . . .	31
2.2	DSM calculation. If we take $\vec{z} = \vec{AB}$ as a (degenerate) canonical analogy vector, then candidate analogy responses are scored based on the criteria $S_{dsm} = \alpha D_C(\vec{p} - \hat{D}, \vec{z}) + D_C(\vec{p}, \hat{D}) = \alpha(1 - \cos(\phi)) + (1 - \cos(\theta))$ . Inset: DSM search pattern. Rather than selecting candidate points based solely on proximity, DSM searches in expanding rings that are elongated in the direction of the analogy vector. . . . .	33
2.3	Algorithm performance on a sequence of analogical reasoning tasks, showing the % of queries for which the algorithm's <i>first</i> response word was the correct answer. Algorithms utilizing a canonical analogy vector are dramatically superior to the naive application of vector $\vec{AB}$ to $\vec{C}$ . Directional scoring improves performance even further. . . . .	36

2.4	Algorithm performance on the Google Analogy Test Set, showing the % of queries for which the algorithm’s <i>first</i> response word was the correct answer. On these primarily syntactic datasets, it is more difficult to identify a superior algorithm. . . . .	37
2.5	Query responses calculated using the <i>travel</i> dataset. . . . .	40
2.6	Query responses calculated using the <i>locations for objects</i> dataset. . . . .	40
2.7	Query responses calculated using the <i>trash vs. treasure</i> dataset. . . . .	41
3.1	Sample regexv expansions obtained using the 500,000 most common tokens in the English-language fastText corpus. Matches were selected using the neighbor expansion method with $\epsilon=0.35$ . . . . .	46
3.2	Comparison between regex and regexv. Here, [taxi] is shorthand for the intent that would include all words that are synonymous with taxi. . . . .	50
4.1	Sample canonical analogy set used find goal-driven actions. The full set in our experiments included 17 goal-verb pairs and 18 goal-object pairs. . . . .	55
4.2	Agent architecture. State descriptions and human guidance, encoded as skip-thought vectors, are used to influence final action selections. . . . .	57
4.3	Object prioritization based on human guidance. Candidate objects are projected onto the goal vector $G_{noun}$ and ranked based on magnitude. The system then identifies a set of candidate verbs that are compatible with the chosen object and prioritizes them with respect to $G_{verb}$ . . . . .	58
4.4	The human-defined goals for each game, reflecting high-level knowledge of the reward structure. Critically, none of the goal texts produce rewards in their own right; instead, they provide guidance that the agent uses to select promising noun-verb combinations. . . . .	59

4.5	<b>cyan:</b> random actions. <b>green:</b> no human guidance. <b>red:</b> human guidance. Clustered bars show changes in game score as the agent is permitted to try [30,15,3,1] verbs for each object. Agents were allowed to interact with each game for 2000 time steps, and the results from 45 data runs were averaged. . . . .	60
5.1	Example texts used to define the ‘Threat’ mode, meaning that an immediate physical danger is present. . . . .	67
5.2	Top: Classification accuracy, counting each image/category pair individually. Bottom: Exact matches, meaning the percentage of images where all four categories were classified correctly. Human captions were not available for the full dataset, and so are not shown here. . . . .	70
5.3	Overall AEGIM correctly identifies exploration scenes regardless of captioning method, perhaps because Clarifai and CaptionBot both detect landscape elements like hills, clouds, and buildings. . . . .	71
5.4	This is one of the few combat images that was even moderately well-captioned by the automated systems. (Scenes with similar elements were described as ‘A group of people jumping’ or ‘a young man practicing his tricks on his skateboard’) . . . . .	72
5.5	In this case the human captioner’s commentary on the shopkeeper’s physique obscured the correct classification. AEGIM correctly returns the interaction mode ‘Barter’ when given the input text ‘A blacksmith sitting in his shop’. . . . .	73
5.6	It is not difficult to see why CaptionBot mistook the image for a clock. Less obvious is why AEGIM does not consider the clock a puzzle item to be interacted with. . . . .	74

6.1	Conversational Scaffolding workflow: The dialog context is converted to an array of sentence-level linguistic embeddings using Google’s Universal Sentence Encoder, then passed to one of several localization methods in order to determine which utterances in the scaffold corpus represent the best context match. The matching contexts (orange), along with the embedded representation of their direct successors (red) within the scaffold corpus, are then passed to the Response Scoring Algorithm, which assigns a numerical value to each candidate response. . . . .	81
6.2	Naive Analogy, where: $c$ (green) represents the embedded input utterance, $a_i$ (blue) represent the nearest embedded utterances from the scaffold corpus, $b_i$ (red) represent the embedded successors to $a_i$ in the scaffold corpus, $d_1 = c + b_2 - a_2$ (yellow) represents the ‘ideal’ response, and $g_i$ (grey and black) represent embedded candidate responses with $g_1$ (black) representing the response selected by the naive-analogy scoring algorithm. . . . .	83
6.3	Scattershot Method, where: $c$ (green) represents the embedded input utterance, $a_i$ (blue) represent the nearest embedded utterances from the scaffold corpus, $b_i$ (red) represent the associated embedded successors to $a_i$ in the scaffold, $d_i = c + (b_i - a_i)$ (yellow) represent the ‘ideal’ responses, and $g_i$ (grey and black) represent embedded candidate responses with $g_3$ (black) representing the response selected by the scattershot scoring algorithm. . . . .	86
6.4	Flow Vectors Method, where: $c$ (green) represents the embedded input utterance, $a_i$ (blue) represent the nearest embedded utterances from the scaffold corpus, $b_i$ (red) represent the associated embedded successors to $a_i$ in the scaffold corpus, $d_1 = c + 1/n \sum(b_i - a_i)$ (yellow) represents the ‘ideal’ response, and $g_i$ (grey and black) represent embedded candidate responses with $g_2$ (black) representing the response selected by the flow vectors scoring algorithm. . . . .	86

7.1	Our architecture is built on the premise of multiple candidate response generators vying for the attention of the dialog manager. Each generator produces a specific type of response: emotive utterance, fact retrieval, conversation offer, self-analysis, etc. The dialog manager analyzes these outputs, then selects and combines the best of them to produce a fluid, natural-sounding response. . . .	99
7.2	A representation of a portion of our knowledge graph. Orange nodes and edges correspond to general facts contained in the knowledge graph, the green node and edges represent a model of the user associated with a specific userID, and the blue node and edge represent a portion of the system model corresponding to the same userID. Bolded edge labels represent facts of interest that elicit responses from the MCI and MCE algorithms discussed in Section 7.3.1. . . .	102
7.3	Naive Offset, where: $c$ (green) represents the embedded utterance, $a_i$ (blue) represent the nearest embedded utterances from the Chit-Chat Dataset, $b_i$ (red) represent the associated embedded response to $a_i$ in the Chit-Chat Dataset, $d_1$ (yellow) represents the ‘ideal’ response, and $g_i$ (grey and black) represent embedded responses generated by our system with $g_1$ (black) representing the response selected by the Naive ranking strategy. . . . .	116
7.4	Scattershot Method, where: $c$ (green) represents the embedded utterance, $a_i$ (blue) represent the nearest embedded utterances from the Chit-Chat Dataset, $b_i$ (red) represent the associated embedded response to $a_i$ in the Chit-Chat Dataset, $d_i$ (yellow) represent the ‘ideal’ responses, and $g_i$ (grey and black) represent embedded responses generated by our system with $g_3$ (black) representing the response selected by the Scattershot ranking strategy.	117



7.5	Flow Vectors Method, where: $c$ (green) represents the embedded utterance, $a_i$ (blue) represent the nearest embedded utterances from the Chit-Chat Dataset, $b_i$ (red) represent the associated embedded response to $a_i$ in the Chit-Chat Dataset, $d_1$ (yellow) represents the ‘ideal’ response, and $g_i$ (grey and black) represent embedded responses generated by our system with $g_2$ (black) representing the response selected by the Flow Vectors ranking strategy.	118
7.6	A list of the created intents and a brief description of each one. . . . .	124
7.7	A list of user intents and response intents paired to indicate either a positive or negative match. . . . .	125
7.8	A list of the features and their respective scalars. . . . .	126
8.1	Screen Shot from FloydHub.com [63], outlining an automated customer support algorithm that relies heavily on the semantic structure of a pre-trained embedding space. The proposed algorithm is apt to deliver wildly inappropriate responses whenever the user’s utterance is semantically distinct from - but contextually similar to - one of its previously-saved questions. . . . .	135
8.2	Our neural model. Pre-trained FastText embeddings are fed into a GRU encoder, with the averaged hidden states passed to a single, fully-connected layer. Although the network architecture is straightforward, the training tasks used allow it to perform comparably with state-of-the-art models on the SemEval 2013 benchmark, while outperforming them on two semantic discrimination tasks. The model is relatively lightweight, with only 4 layers and approximately 31 million trainable weights. The output embedding is a 300-dimensional vector.	140
8.3	Sample sentence trios from the <i>Negation</i> and <i>Inversion</i> evaluation sets. . . .	141

8.4	Categorization accuracy on a variety of semantic similarity and common-sense reasoning tasks. <b>SemEval:</b> A sentiment analysis task requiring the model to predict the human classification of sentences; <b>Inv:</b> An inversion task requiring discernment between two sentences in which some words have been swapped for antonyms; <b>Neg:</b> A negation task requiring discernment between two sentences that differ only by inclusion of the word ‘not’; <b>Avg:</b> The average of the values in the first three columns. The highest accuracy in each column is bolded. A box has been drawn around the model that does the best job of maximizing average score without compromising performance at SemEval. . . . .	143
8.5	cosine distances between sentences embedded using various neural models. Each sentence pair comprises two rows, with distances shown after the second component sentence. 4-tuples that evidence semantically appropriate distances are bolded. . . . .	145

## List of Tables

2.1	Analogy solutions provided by traditional offset methods and DSM when using a canonical averaged analogy vector. The first response provided by each algorithm is shown alongside the number of plausible responses returned (out of a total of 15). Entries are bolded if either the first response was valid or the number of valid responses was greatest. POS tags have been omitted for clarity. *In these cases, the method returned a differently-tagged version of the source word. . . . .	42
3.1	Cosine distance between single-word vectors and compositional vectors created by averaging the elements of a multi-word phrase. The resulting distances correlate with many human estimations of semantic similarity. . . . .	50
6.1	Dialog localization scores obtained using the dialog history and possible contextual matches shown in Section 3.1. Lower scores (bolded) indicate a better contextual match. . . . .	84
6.2	Dataset Statistics . . . . .	88
6.3	Algorithm accuracy on a response prioritization task with 13,244 distinct conversations. The value $n$ denotes the length of the dialog history used. The neural network was unable to select a response in some contexts because it requires two distinct vectors as input. The highest-scoring algorithm in each column is shown in bold-face text. . . . .	90

6.4	Confusion matrix showing how well each dataset, when used as a scaffold corpus, is able to select appropriate responses for dialogues drawn from the other corpora. Each column contains a scaffold corpus, each row an evaluation corpus. The scattershot algorithm was used in conjunction with the embedded concatenation localization method, with a dialog history of size $n = 2$ and with 3,311 evaluation dialogues drawn from each corpus. The highest accuracy level in each column is shown in bold-face text. . . . .	92
7.1	Query: “Where are you from?” . . . . .	119
7.2	Query: “Where were you born?” . . . . .	119
7.3	Top: Accuracy of scaffolding algorithms in offline tests. To count as correct, the ideal answer must be in the top 40% of the algorithm’s preferred responses. Bottom: Average rating received during A/B testing. Tests were performed between 8/18/2018 and 8/24/2018 inclusive, with each conversation randomly assigned to use one of the scaffolding variants, the neural network predictor, or no scaffolding algorithm at all. All other system parameters were held constant.	120

## Part I

### Overview

This dissertation is all about natural language understanding. As humans, we communicate with one another through simple verbal expressions. Other humans are able to interpret and act upon these expressions almost flawlessly, even when the original statement was ambiguous or incomplete, and yet imbuing computational systems with the same abilities has proved a challenging task. As a society, we yearn for Jarvis, KIT, the Star Trek Computer... Instead we get Alexa, Cortana and Siri.

How do we bridge that gap? We want to create systems that are able to understand language and reason with it, but how do you comprehend the meaning of words like “touch”, “sky”, and “ground” when you have neither hands, eyes, nor feet?

A solution is expressed in the Distributional Hypothesis of linguistics, which states that words which appear in the same contexts tend to have similar meanings [58]. Thus, the meaning of a word can be defined in terms of its relationships with other words, independent of its grounding (or lack thereof) in real-world experience. This concept is the basis of recent popular embedding models such as word2vec, GLoVE, and fastText [11, 97, 117], which use the context of a word, or “the company it keeps” [39] to create numerical representations of language. In the scientific literature, these representations are described interchangeably as ‘distributed representations’, ‘linguistic embeddings’, or ‘word vectors’.

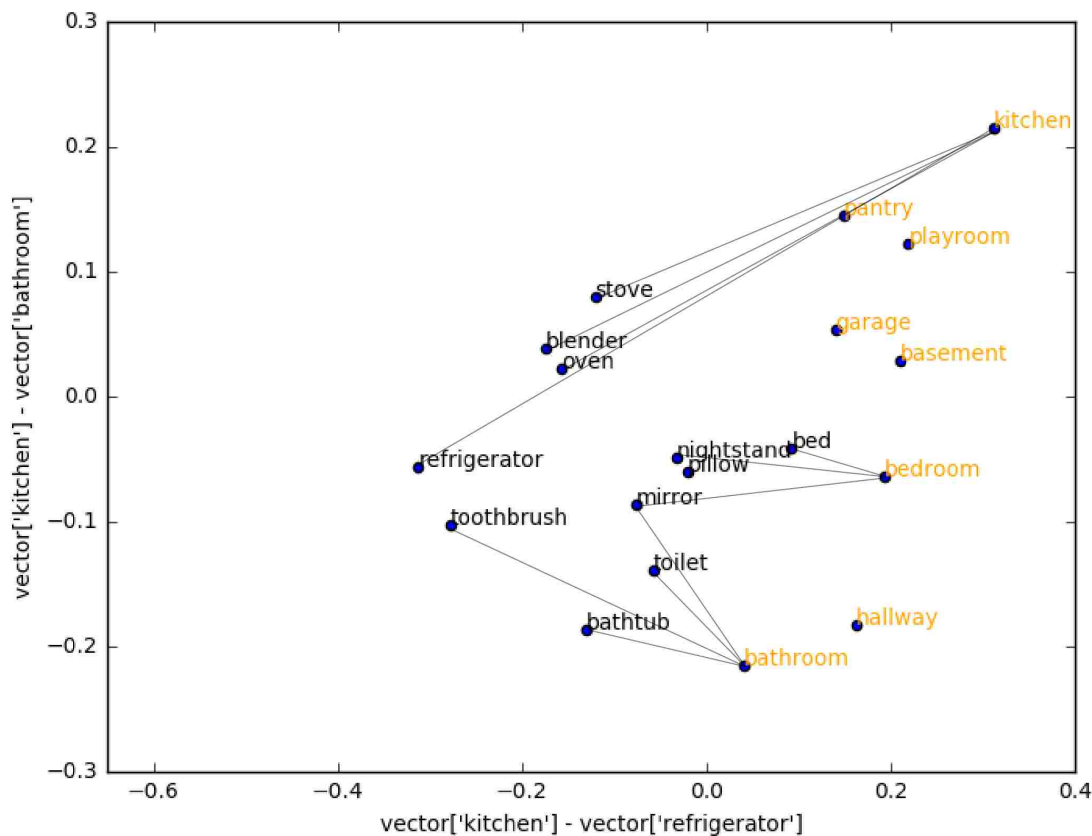


Figure 1: 100-dimensional word vectors projected into a 2-dimensional space demonstrate impressive semantic structure. Common household items are clustered in specific relationships to the rooms in which they tend to be located. The angle of correspondence is at least as significant as proximity. Of particular interest is the word ‘mirror’, which is aligned both with the items found in a bedroom and with the items found in a bathroom.

Experiments with these embeddings reveal that they have surprising and powerful properties. For example, words that describe common household objects are positioned in close proximity to objects that tend to appear within the same room. For example, common appliances like ‘stove’, ‘refrigerator’, and ‘blender’ lie close to one another and share a common angle of incidence toward the representation for ‘kitchen’ (see Figure 1).

Perhaps even more startling, the embedding spaces are structured such that words which are related to one another in specific ways map to locations with matching geometric properties. For example, the difference between the vectors for ‘king’ and ‘queen’ is almost

identical to the vector difference between ‘man’ and ‘woman’, suggesting that a form of everyday common-sense knowledge is implicitly encoded within the structure of the embedding space. It ‘knows’, in some sense, that a queen is a female ruler, just as it ‘knows’ that an oven can be found in a kitchen.

A major contribution of this dissertation is the idea that common-sense knowledge can be found directly within linguistic embeddings, and that this knowledge can be extracted and utilized for real-world applications like robot navigation and dialog systems. We further demonstrate a method for training embedding spaces with even stronger structural properties.

## **Linguistic Embedding Models**

Linguistic embeddings, also known as vector space models or distributed representations of words, rose to unprecedented prominence within the research community with the introduction of the GLoVE [116] and word2vec [97] algorithms, both of which use unsupervised methods to produce vector representations of words. Additional word-level embedding models followed, including the FastText algorithm [11], which utilizes subword information to enrich the resulting word vectors, and contextualized word embeddings such as those produced by BERT [30].

Many of these embedding models encode the semantic meaning of words so precisely that simple algebra followed by a nearest-neighbor search is sufficient to solve analogical queries such as man:woman::king:queen (‘man is to woman as king is to queen’). This prompted researchers to explore the natural extension of word-level linguistic models to the level of sentences and paragraphs. In 2016, Kiros et al. presented skip-thoughts [75], an extension to multi-word text of the context-prediction task used to train word2vec. Two years later Google released its Universal Sentence Encoder [16], which features two variants: A lightweight implementation that disregards syntax in favor of a quickly trainable bag-of-words representation [65] and a large model based on a Transformer architecture structured around attention mechanisms [157]. Trained on a combination of tasks including context prediction;

an input-response task; and classification tasks using supervised data, Google’s encoder became the de facto standard for use as pre-trained language model input features until its displacement by BERT later that year.

Unfortunately, the structural properties of word-level embedding spaces fail to manifest effectively in their sentence-level extensions. For example, skip-thought embeddings place otherwise identical sentences in wildly disparate locations based on ending punctuation and both of Google’s universal encoder models map negation pairs such as “I am a cat” and “I am not a cat” to nearly identical vectors (see Chapter 8). The models were not optimized for linguistic reasoning tasks, and their structural irregularities prevent them from being used in the same way that word embeddings are applied.

Perhaps for this reason, many applications of pre-trained linguistic embedding models use them as inputs for end-to-end tasks such as dialog modeling, machine translation, and question answering. In such contexts, it is not troubling that skip-thought embeddings are more sensitive to syntax than to semantics, or that Google’s universal encoder distinguishes only subtly between a sentence and its negation. After all, subsequent layers of the neural network can smooth over these irregularities and produce the desired result [75].

This research addresses the gap between the state of current embedding spaces and the need for general-purpose linguistic knowledge by presenting a novel, carefully-curated network architecture designed for the explicit purpose of reasoning in the linguistic domain. Rather than using our model to produce pre-trained input features for downstream tasks, we instead focus on common-sense reasoning via algebraic operations performed directly within the embedding space, without the need for additional training.

This somewhat unorthodox application mirrors the way linguistic models are often used “in the wild”, by developers and hobbyists. By and large, when one reads blog posts about word embeddings or browses through discussion forums, one notices an inherent fascination with the embeddings themselves, independent of any transfer learning they facilitate. A plethora of web sites have popped up allowing users to experiment with embedding spaces by



projecting words into interesting subspaces and observing the patterns they generate. The vector properties in the python spacy module are frequently used as an analogue for semantic meaning, with dot products between vectors used as a measure of similarity. The average user seems to want to use the linguistic embeddings *directly* - and there is a certain intuitive appeal to this approach. After all, if words and sentences can be represented as numbers, should one not be able to manipulate them mathematically?

## Doing Math with Language

The answer, of course, is that one *can*, at least at the level of single words. In 2013, Mikolov et al. observed what has since become a hallmark feature of word-level embedding models: their ability to represent linguistic regularities in the form of analogical relations [100]. Although trained for different purposes entirely, most word-level embedding models can be used to solve analogical queries of the form  $a:b::c:d$  ( $a$  is to  $b$  as  $c$  is to  $d$ ). Leveraging this principle, it is possible to use simple linear relations to discover the unknown value  $d$ , e.g.

$$\text{Madrid} - \text{Spain} + \text{France} = \text{Paris}$$

The possibilities are tantalizing. For example, researchers have demonstrated that mathematical operations on word embeddings can be used to detect affordances [45], infer the locations of everyday objects [48], and condition agent behaviors on natural language instructions [44].

A natural extension would be to apply these same principles at the sentence level - and yet there is a notable dearth of papers demonstrating such applications, primarily because coherent results are more difficult to achieve. For example, using a pre-trained skip-thought encoder [72] and corresponding decoder trained on reddit data, the following equivalences hold:

$$\text{'I am angry'} - \text{'angry'} + \text{'happy'} = \text{'I happy'}$$

$$\text{'thank you'} + \text{'you're welcome'} = \text{'enjoy you'}$$

At first glance, it appears that this sentence-level embedding space is functioning with the same precision as its word-level predecessors. Alas, the illusion breaks down as soon as more equations are attempted:

$$\textit{‘a dragon flying’} + \textit{‘high in the sky’} = \textit{‘a high’}$$

$$\textit{‘the sky is blue’} - \textit{‘blue’} + \textit{‘orange’} = \textit{‘the orange is the orange’}$$

Clearly the *potential* for direct mathematical manipulation of the space is present, but the space is insufficiently structured (or the decoder insufficiently trained) to allow this information to be reliably extracted.

## Reasoning in the Linguistic Domain

When we speak of algorithm-based reasoning, we generally imagine a system that transforms language or other observations into symbols and then manipulates the symbols according to a set of predefined rules in order to achieve the desired result.

This work lays the foundation for future applications that “reason in the linguistic domain”. We mean by this a system that reasons via mathematical traversals of a linguistic embedding space. While this process could be formalized within the framework of traditional reasoning (with vector representations corresponding to symbols and mathematical operations corresponding to symbol manipulation rules) we find it distinct in that it is simultaneously less precise and more flexible than traditional reasoning frameworks.

Work done for this dissertation shows, for example, that this method can be used to determine the best mode of transportation to specific locations (see Chapter 2) and to detect potential threats based on a verbal description of a scene (see Chapter 5). It is our hope that as the precision and representational capacities of linguistic embedding spaces improve, the pathway will open for new applications related to question answering, dialog modeling, and conversational AI.

## **Research Objectives/Paper Overview**

The purpose of this research is twofold: First, to demonstrate the powerful potential of linguistic embedding models as common-sense knowledge repositories. Second, to develop an embedding model that excels at such applications.

The remainder of this work is structured as follows: Chapters 1-4 present research into common-sense applications of linguistic embeddings comprising only single words or short phrases. We show that word and phrase embeddings can be used to detect affordances, solve high-level robot navigation tasks, expand regex statements in semantically meaningful ways, and allow agents to act in compliance with natural language instructions. Chapters 5-7 present research that applies the same methodologies to sentence-level reasoning tasks. The results are intriguing yet suboptimal. Chapter 8 presents a deep constrained neural embedding model that is able to capture critical semantic structure, such as sentence negations, in the geometry of the resulting embedding space. We show that this model is able to outperform state-of-the-art embeddings on two semantic discernment tasks.

## Part II

### The Quest for Common-Sense Knowledge

Common-sense knowledge is intrinsic to our perception of the world. We see a closed door and instantly understand that a new environment lies beyond it. We see gyrating reflections and immediately know we are looking at a body of water. Common-sense knowledge also helps us to make predictions: A dropped ball will bounce. A tipped glass will spill its contents. Perhaps most importantly, we rely on common-sense knowledge to resolve ambiguities in spoken language. Given the phrase ‘Let’s meet at three’, we determine from context that the word ‘three’ refers to a time instead of a location or a numeral. We can also infer that the speaker intends to meet at 3:00 in the afternoon rather than at 3:00 in the morning.

These and similar experiences are so ubiquitous that we seldom notice the assumptions we make or the way our expectations shape our experience. And yet common-sense reasoning is seldom incorporated into (or extracted from) machine learning algorithms. Neural networks operate on the assumption that there exist direct, unequivocal mappings between observed inputs and the optimal output. Even probabilistic models such as variational autoencoders [70] do not represent common-sense knowledge explicitly; instead these models attempt to learn probability distributions for the unknown latent variables that drive the output, without any understanding of how those variables relate to its experience or to the world in general.

The chapters in this section validate an alternate view, showing that linguistic embeddings can be treated as a common-sense knowledge repository that implicitly models our

everyday word by placing words with similar meanings in proximity to one another. Further, it is shown that the embeddings for related words are encoded in ways that preserve those relationships.

*Chapter 1: What Can You Do With a Rock? Affordance Extraction via Word Embeddings* leverages linguistic relationships to improve agent performance on text-based adventure games, a challenging learning environment with a dauntingly large action space. *Chapter 2: Harvesting Common-sense Navigational Knowledge for Robotics from Uncurated Text Corpora* extends these ideas further, showing that the word2vec embedding space models information about the world as a learned geometry of the contexts in which human beings tend to use each idea. *Chapter 3: Embedding Grammars* presents some preliminary explorations on a combination of word embeddings with context-free grammars, thus allowing classical regex expressions to match a broader spectrum of terms than those entered by the programmer. Finally, *Chapter 4: Informing Action Primitives Through Free-Form Text* uses embedded sentence representations to prioritize the action primitives of an agent interacting with a simulated environment, thus allowing the agent to select actions that conform with natural-language instructions provided by a human. A key advantage of this approach is its flexibility: once a properly-structured embedding space has been obtained, new mappings between instructions and behaviors can be formed at will, without the need for additional training.

Although this work focuses on text-based representations of knowledge, we note that there are many other ways to model world data. Maps, images, behavioral skills, statistical models and physics engines all represent forms of common-sense knowledge that can be leveraged within autonomous systems. While we value these approaches, we have restricted our current attention to text-based embedding spaces.

## Chapter 1

### What Can You Do with a Rock?

#### Affordance Extraction via Word Embeddings

by Nancy Fulda, Daniel Ricks, Ben Murdoch and David Wingate,  
International Joint Conference on Artificial Intelligence (IJCAI 2017).

#### Abstract

Autonomous agents must often detect *affordances*: the set of behaviors enabled by a situation. Affordance detection is particularly helpful in domains with large action spaces, allowing the agent to prune its search space by avoiding futile behaviors. This paper presents a method for affordance extraction via word embeddings trained on a tagged Wikipedia corpus. The resulting word vectors are treated as a common knowledge database which can be queried using linear algebra. We apply this method to a reinforcement learning agent in a text-only environment and show that affordance-based action selection improves performance in most cases. Our method increases the computational complexity of each learning step but significantly reduces the total number of steps needed. In addition, the agent's action selections begin to resemble those a human would choose.

#### 1.1 Introduction

The physical world is filled with constraints. You can open a door, but only if it isn't locked. You can douse a fire, but only if a fire is present. You can throw a rock or drop a rock or even, under certain circumstances, converse with a rock, but you cannot traverse it, enumerate

it, or impeach it. The term *affordances* [53] refers to the subset of possible actions which are feasible in a given situation. Human beings detect these affordances automatically, often subconsciously, but it is not uncommon for autonomous learning agents to attempt impossible or even ridiculous actions, thus wasting effort on futile behaviors.

This paper presents a method for affordance extraction based on the copiously available linguistic information in online corpora. Word embeddings trained using Wikipedia articles are treated as a common sense knowledge base that encodes (among other things) object-specific affordances. Because knowledge is represented as vectors, the knowledge base can be queried using linear algebra. This somewhat counterintuitive notion - the idea that words can be manipulated mathematically - creates a theoretical bridge between the frustrating realities of real-world systems and the immense wealth of common sense knowledge implicitly encoded in online corpora.

We apply our technique to a text-based environment and show that *a priori* knowledge provided by affordance extraction greatly speeds learning. Specifically, we reduce the agent’s search space by (a) identifying actions afforded by a given object; and (b) discriminating objects that can be grasped, lifted and manipulated from objects which can merely be observed. Because the agent explores only those actions which ‘make sense’, it is able to discover valuable behaviors more quickly than a comparable agent using a brute force approach. Critically, the affordance agent is demonstrably able to eliminate extraneous actions without (in most cases) discarding beneficial ones.

All the experiments in this paper were performed using a Wikipedia corpus that had been preprocessed to append part-of-speech tagging to each word.

## 1.2 Related Work

Our research relies heavily on word2vec [96], an algorithm that encodes individual words based on the contexts in which they tend to appear. Earlier work has shown that word vectors trained using this method contain intriguing semantic properties, including structured representations

of gender and geography [99, 101]. The (by now) archetypal example of such properties is represented by the algebraic expression  $vector[‘king’] - vector[‘man’] + vector[‘woman’] = vector[‘queen’]$ .

Researchers have leveraged these properties for diverse applications including sentence- and paragraph-level encoding [74, 82], image categorization [42], bidirectional retrieval [68], semantic segmentation [137], biomedical document retrieval [15], and the alignment of movie scripts to their corresponding source texts [175]. Our work is most similar to [173]; however, rather than using a Markov Logic Network to build an explicit knowledge base, we instead rely on the semantic structure implicitly encoded in skip-grams.

Affordance detection, a topic of rising importance in our increasingly technological society, has been attempted and/or accomplished using visual characteristics [138, 139], haptic data [107], visuomotor simulation [132, 133], repeated real-world experimentation [105, 141], and knowledge base representations [173].

In 2001 [80] identified text-based adventure games as a step toward general problem solving. The same year at AAAI, Mark DePristo and Robert Zubek unveiled a hybrid system for text-based game play [4], which operated on hand-crafted logic trees combined with a secondary sensory system used for goal selection. The handcrafted logic worked well, but goal selection broke down and became cluttered due to the scale of the environment. Perhaps most notably, in 2015 [106] designed an agent which passed the text output of the game through an LSTM [61] to find a state representation, then used a DQN [104] to select a Q-valued action. This approach appeared to work well within a small discrete environment with reliable state action pairs, but as the complexity and alphabet of the environment grew, the clarity of Q-values broke down and left them with a negative overall reward. Our work, in contrast, is able to find meaningful state action pairs even in complex environments with many possible actions.



### 1.3 Wikipedia as a Common Sense Knowledge Base

Google ‘knowledge base’, and you’ll get a list of hand-crafted systems, both commercial and academic, with strict constraints on encoding methods. These highly-structured, often node-based solutions are successful at a wide variety of tasks including topic gisting [87], affordance detection [173] and general reasoning [127]. Traditional knowledge bases are human-interpretable, closely tied to high-level human cognitive functions, and able to encode complex relationships compactly and effectively.

It may seem strange, then, to treat Wikipedia as a knowledge base. When compared with curated solutions like ConceptNet [87], Cyc [95], and WordNet [102], its contents are largely unstructured, polluted by irrelevant data, and prone to user error. When used as a training corpus for the word2vec algorithm, however, Wikipedia becomes more tractable. The word vectors create a compact representation of the knowledge base and, as observed by [12] and [13], can even encode relationships about which a human author is not consciously cognizant. Perhaps most notably, Wikipedia and other online corpora are constantly updated in response to new developments and new human insight; hence, they do not require explicit maintenance.

However: in order to leverage the semantic structure implicitly encoded within Wikipedia, we must be able to interpret the resulting word vectors. Significant semantic relationships are not readily apparent from the raw word vectors or from their PCA reduction. In order to extract useful information, the database must be queried through a mathematical process. For example, in Figure 1.1 a dot product is used to project gendered terms onto the space defined by  $vector[‘king’] - vector[‘queen’]$  and  $vector[‘woman’] - vector[‘man’]$ . In such a projection, the mathematical relationship between the words is readily apparent. Masculine and feminine terms become linearly separable, making it easy to distinguish instances of each group.

These relationships can be leveraged to detect affordances, and thus reduce the agent’s search space. In its most general interpretation, the adjective *affordant* describes the set of

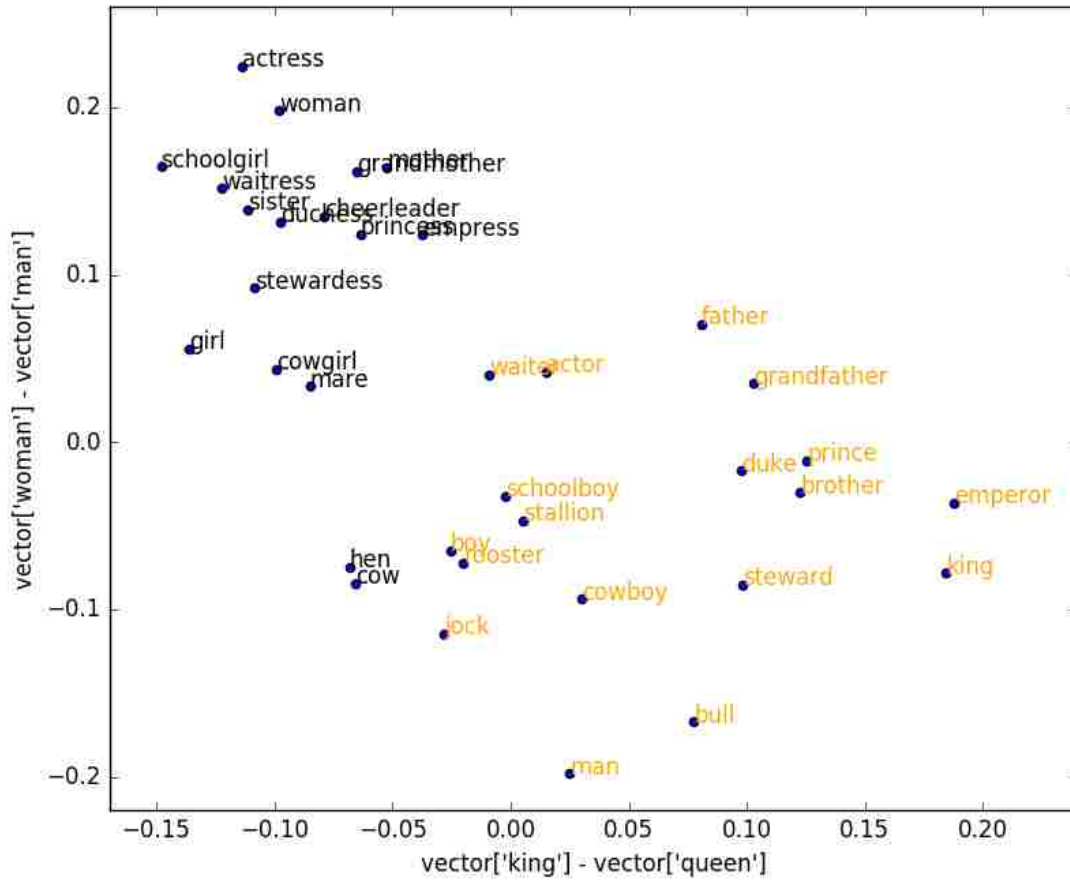


Figure 1.1: Word vectors projected into the space defined by  $vector[‘king’] - vector[‘queen’]$  and  $vector[‘woman’] - vector[‘man’]$ . In this projection, masculine and feminine terms are linearly separable.

actions which are **physically possible** under given conditions. In the following subsections, however, we use it in the more restricted sense of actions which **seem reasonable**. For example, it is physically possible to eat a pencil, but it does not ‘make sense’ to do so.

### 1.3.1 Verb/Noun Affordances

So how do you teach an algorithm what ‘makes sense’? We address this challenge through an example-based query. First we provide a canonical set of verb/noun pairs which illustrate the relationship we desire to extract from the knowledge base. Then we query the database using the analogy format presented by [96]. Using their terminology, the analogy `sing:song::[:?]:[x]` encodes the following question: If the affordant verb for ‘song’ is ‘sing’, then what is the affordant verb for [x]?

In theory, a single canonical example is sufficient to perform a query. However, experience has shown that results are better when multiple canonical values are averaged.

More formally, let  $W$  be the set of all English-language word vectors in our agent’s vocabulary. Further, let  $N = \{\vec{n}_1, \dots, \vec{n}_j\}, N \subset W$  be the set of all nouns in  $W$  and let  $V = \{\vec{v}_1, \dots, \vec{v}_k\}, V \subset W$  be the set of all verbs in  $W$ .

Let  $C = \{(\vec{v}_1, \vec{n}_1), \dots, (\vec{v}_m, \vec{n}_m)\}$  represent a set of canonical verb/noun pairs used by our algorithm. We use  $C$  to define an affordance vector  $\vec{a} = 1/m \sum_i (\vec{v}_i - \vec{n}_i)$ , which can be thought of as the distance and direction within the embedding space which encodes affordant behavior.

In our experiments we used the following verb/noun pairs as our canonical set:

[‘sing song’, ‘drink water’, ‘read book’, ‘eat food’, ‘wear coat’,  
‘drive car’, ‘ride horse’, ‘give gift’, ‘attack enemy’, ‘say word’,  
‘open door’, ‘climb tree’, ‘heal wound’, ‘cure disease’,  
‘paint picture’]

Our algorithm		Co-occurrence		Concept Net	
vanquish	impale	have	die	kill	harm
duel	battle	make	cut	parry	fence
unsheath	behead	kill	fight	strike	thrust
summon	wield	move	use	slash	injure
overpower	cloak	destroy	be	look cool	cut

Figure 1.2: Verb associations for the noun ‘sword’ using three different methods: (1) Affordance detection using word vectors extracted from Wikipedia, as described in this section, (2) Strict co-occurrence counts using a Wikipedia corpus and a co-occurrence window of 9 words, (3) Results generated using ConceptNet’s CapableOf relationship.

We describe a verb/noun pair  $(\vec{v}, \vec{n})$  as affordant to the extent that  $\vec{n} + \vec{a} = \vec{v}$ . Therefore, a typical knowledge base query would return the  $n$  closest verbs  $\{\vec{v}_{c1}, \dots, \vec{v}_{cn}\}$  to the point  $\vec{n} + \vec{a}$

For example, using the canonical set listed above and a set of pre-trained word vectors, a query using  $\vec{n} = \text{vector}[\text{‘sword’}]$  returns the following:

[‘vanquish’, ‘duel’, ‘unsheathe’, ‘wield’, ‘summon’, ‘behead’,  
‘battle’, ‘impale’, ‘overpower’, ‘cloak’]

Intuitively, this query process produces verbs which answer the question, ‘What should you do with an [x]?’. For example, when word vectors are trained on a Wikipedia corpus with part-of-speech tagging, the five most affordant verbs to the noun ‘horse’ are {‘gallop’, ‘ride’, ‘race’, ‘horse’, ‘outrun’}, and the top five results for ‘king’ are {‘dethrone’, ‘disobey’, ‘depose’, ‘reign’, ‘abdicate’}.

The resulting lists are surprisingly logical, especially given the unstructured nature of the Wikipedia corpus from which the vector embeddings were extracted. Subjective examination suggests that affordances extracted using Wikipedia are at least as relevant as those produced by more traditional methods (see Figure 1.2).

It is worth noting that our algorithm is not resilient to polysemy, and behaves unpredictably when multiple interpretations exist for a given word. For example, the verb ‘eat’ is highly affordant with respect to most food items, but the twelve most salient results

for ‘apple’ are {‘apple’, ‘package’, ‘program’, ‘release’, ‘sync’, ‘buy’, ‘outsell’, ‘download’, ‘install’, ‘reinstall’, ‘uninstall’, ‘reboot’}. In this case, ‘Apple, the software company’ is more strongly represented in the corpus than ‘apple, the fruit’.

---

**Algorithm 1** Verb Selection With Analogy Reduction

---

- 1: *navigation\_verbs* = [‘north’, ‘south’, ‘east’, ‘west’, ‘northeast’, ‘southeast’, ‘southwest’, ‘northwest’, ‘up’, ‘down’, ‘enter’]
  - 2: *manipulation\_verbs* = a list of 1000 most common verbs
  - 3: *essential\_manipulation\_verbs* = [‘get’, ‘drop’, ‘push’, ‘pull’, ‘open’, ‘close’]
  - 4: *affordant\_verbs* = verbs returned by Word2vec that match *noun*
  - 5: *affordant\_verbs* = *affordant\_verbs*  $\cap$  *manipulation\_verbs*
  - 6: *final\_verbs* = *navigation\_verbs*  $\cup$  *affordant\_verbs*  $\cup$  *essential\_manipulation\_verbs*
  - 7: *verb* = a randomly selected verb from *final\_verbs*
- 

### 1.3.2 Identifying Graspable Objects

Finding a verb that matches a given noun is useful. But an autonomous agent is often confronted with more than one object at a time. How should it determine which object to manipulate, or whether any of the objects are manipulable? Pencils, pillows, and coffee mugs are easy to grasp and lift, but the same cannot be said of shadows, boulders, or holograms.

To identify affordant nouns - i.e. nouns that can be manipulated in a meaningful way - we again utilize analogies based on canonical examples. In this section, we describe a noun as *affordant* to the extent that it can be pushed, pulled, grasped, transported, or transformed. After all, it would not make much sense to lift a sunset or unlock a cliff.

We begin by defining canonical affordance vectors  $\vec{a}_x = \vec{n}_{x1} - \vec{n}_{x2}$  and  $\vec{a}_y = \vec{n}_{y1} - \vec{n}_{y2}$  for each axis of the affordant vector space. Then, for each object  $\vec{o}_i$  under consideration, we generate a pair of projections  $\vec{p}_{o_{ix}} = \vec{o}_i \text{ dot } \vec{a}_x$  and  $\vec{p}_{o_{iy}} = \vec{o}_i \text{ dot } \vec{a}_y$ .

The results of such a projection can be seen in Figure 1.3. This query is distinct from those described in section 2.3.1 because, instead of using analogies to test the relationships between nouns and verbs, we are instead locating a noun on the spectrum defined by two other words.

In our experiments, we used a single canonical vector,  $vector[‘forest’] - vector[‘tree’]$ , to distinguish between nouns of different classes. Potentially affordant nouns were projected

---

**Algorithm 2** Noun Selection With Affordance Detection

---

```
1:  $state =$  game response to last command
2:  $manipulable\_nouns \leftarrow \{\}$ 
3: for each word  $w \in state$  do
4:   if  $w$  is a noun then
5:     if  $w$  is manipulable then
6:       add  $w$  to  $manipulable\_nouns$ 
7:     end if
8:   end if
9: end for
10:  $noun =$  a randomly selected noun from  $manipulable\_nouns$ 
```

---

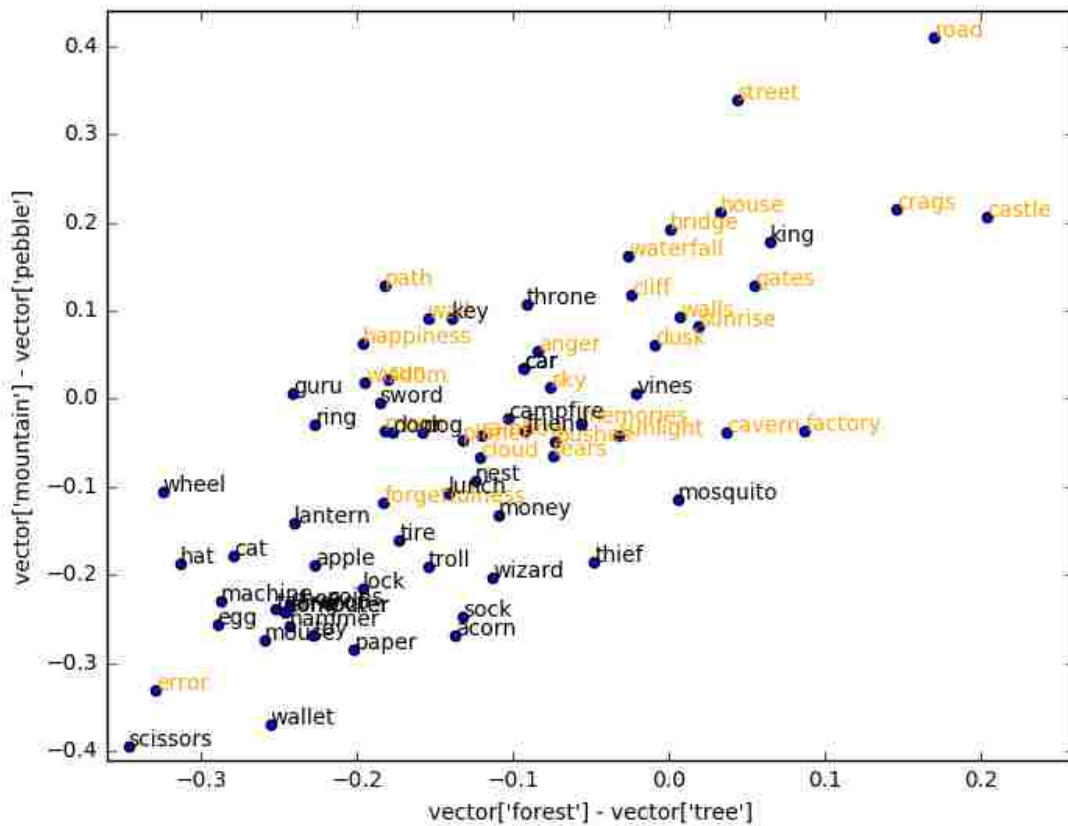


Figure 1.3: Word vectors projected into the space defined by  $vector[‘forest’] - vector[‘tree’]$  and  $vector[‘mountain’] - vector[‘pebble’]$ . Small, manipulable objects appear in the lower-left corner of the graph. Large, abstract, or background objects appear in the upper right. An object’s manipulability can be roughly estimated by measuring its location along either of the defining axes.

onto this line of manipulability, with the words whose projection lay closest to ‘tree’ being selected for further experimentation.

Critical to this approach is the insight that canonical word vectors are most effective when they are thought of as exemplars rather than as descriptors. For example,  $vector[‘forest’] - vector[‘tree’]$  and  $vector[‘building’] - vector[‘brick’]$  function reasonably well as projections for identifying manipulable items, whereas  $vector[‘big’] - vector[‘small’]$  is utterly ineffective.

#### 1.4 Test Environment: A World Made of Words

In this paper, we test our ideas in the challenging world of text-based adventure gaming. Text-based adventure games offer an unrestricted, free-form interface: the player is presented with a block of text describing a situation, and must respond with a written phrase. Typical actions include commands such as: ‘examine wallet’, ‘eat apple’, or ‘light campfire with matches’. The game engine parses this response and produces a new block of text. The resulting interactions, although syntactically simple, provide a fertile research environment for natural language processing and human/computer interaction. Game players must identify objects that are manipulable and apply appropriate actions to those objects in order to make progress.

In these games, the learning agent faces a frustrating dichotomy: its action set must be large enough to accommodate any situation it encounters, and yet each additional action increases the size of its search space. A brute force approach to such scenarios is frequently futile, and yet factorization, function approximation, and other search space reduction techniques bring the risk of data loss. We desire an agent that is able to clearly perceive all its options, and yet applies only that subset which is likely to produce results.

In other words, we want an agent that explores the game world the same way a human does: by trying only those actions that ‘make sense’. In the following sections, we show that affordance-based action selection provides a meaningful first step towards this goal.

```
West of House
You are standing in an open field west of a
white house, with a boarded front door.
There is a small mailbox here.

>Open Mailbox
Opening the small mailbox reveals a leaflet.

>Take leaflet
Taken.

>Read leaflet
"WELCOME TO ZORK!"
```

Figure 1.4: Sample text from the adventure game Zork. Player responses follow a single angle bracket.

### 1.4.1 Learning Algorithm

Our agent utilizes Q-learning [162], a reinforcement learning algorithm which attempts to maximize expected discounted reward. Q-values are updated according to the equation

$$\Delta Q_{\pi}(s, a) = \alpha(R(s, a) + \gamma \max_a Q_{\pi}(s', a) - Q_{\pi}(s, a)) \tag{1.1}$$

where  $Q_{\pi}(s, a)$  is the expected cumulative reward for performing action  $a$  in observed state  $s$  and following the policy  $\pi$  thereafter,  $\alpha$  is the learning rate,  $\gamma$  is the discount factor, and  $s'$  is the new state observation after performing action  $a$ . All experiments in this paper used  $\alpha=0.1$  and  $\gamma=0.95$

The agent’s state representation is encoded as a hash of the text provided by the game engine. Actions are comprised of verb/object pairs:

$$a = v + ' ' + o, v \in V, o \in O \tag{1.2}$$

where  $V$  is the set of all English-language verbs and  $O$  is the set of all English-language nouns. To enable the agent to distinguish between state transitions and merely informational feedback, the agent executes a ‘look’ command every second iteration and assumes that the resulting game text represents its new state. Some games append a summary of actions taken and points earned in response to each ‘look’ command. To prevent this from obfuscating the state space, we stripped all numerals from the game text prior to hashing.



Given that the English language contains at least 20,000 verbs and 100,000 nouns in active use, a naive application of Q-learning is intractable. Some form of action-space reduction must be used. For our baseline comparison, we use an agent with a vocabulary consisting of the 1000 most common verbs in Wikipedia plus an 11-word navigation list and a 6-word essential manipulation list as depicted in Algorithm 2. The navigation list contains words which, by convention, are used to navigate through text-based games. The essential manipulation list contains words which, again by convention, are generally applicable to all in-game objects.

The baseline agent does not use a fixed noun vocabulary. Instead, it extracts nouns from the game text using part-of-speech tags. To facilitate game interactions, the baseline agent augments its noun list using adjectives that precede them. For example, if the game text consisted of ‘You see a red pill and a blue pill’, then the agent’s noun list for that state would be [‘pill’, ‘red pill’, ‘blue pill’]. (And its next action is hopefully ‘swallow red pill’).

In Sections 2.5.1 and 2.5.2 the baseline agent is contrasted with an agent using affordance extraction to reduce its manipulation list from 1000 verbs to a mere 30 verbs for each state, and to reduce its object list to a maximum of 15 nouns per state. We compare our approach to other search space reduction techniques and show that the *a priori* knowledge provided by affordance extraction enables the agent to achieve results which cannot be paralleled through brute force methods. All agents used epsilon-greedy exploration with a decaying epsilon.

The purpose of our research was to test the value of affordance-based search space reduction. Therefore, we did not add augmentations to address some of the more challenging aspects of text-based adventure games. Specifically, the agent maintained no representation of items carried in inventory or of the game score achieved thus far. The agent was also not given the ability to construct prepositional commands such as ‘put book on shelf’ or ‘slay dragon with sword’. Our algorithm does not account for physical or temporal dependencies

in which the possession of an item or the execution of a specific action in the past affect the outcome of the current behaviors.

## 1.5 Results

We tested our agent on a suite of 50 text-based adventure games compatible with Infocom’s Z-machine. These games represent a wide variety of situations, ranging from business scenarios like ‘Detective’ to complex fictional worlds like ‘Zork: The Underground Empire’. The core objective of the games vary widely. In some games, the player’s goal is simply to survive, while in others specific items must be acquired. Significantly, the games provide little or no information about the agent’s goals or which actions might provide reward.

During training, the agent interacted with the game engine for 1000 epochs with 1000 training steps per epoch. On each training step, the agent received a reward corresponding to the change in game score. After each epoch the game was restarted and the game score reset, but the agent retained its learned Q-values. Some games, notably zork3, allowed the player to ‘die’ as play advanced, triggering a negative reward and prematurely terminating the training epoch.

One game experienced a scoring malfunction and was excluded from subsequent analysis; however, our verb-space reduction method obtained a score identical to the baseline agent’s in the discarded partial results.

Our affordance-based search space reduction algorithms enabled the agent to score points on 18/49 games, with a peak performance (expressed as a percentage of possible game score) of 57.44% for verb space reduction, 23.90% for object space reduction, and 56.84% when both methods were combined. The baseline agent (see Sec. 4.1) scored points on 12/49 games, with a peak performance of 20.84%. Peak performance is defined as the maximum score achieved over all epochs, a metric that expresses the agent’s ability to comb through the search space and discover areas of high reward.

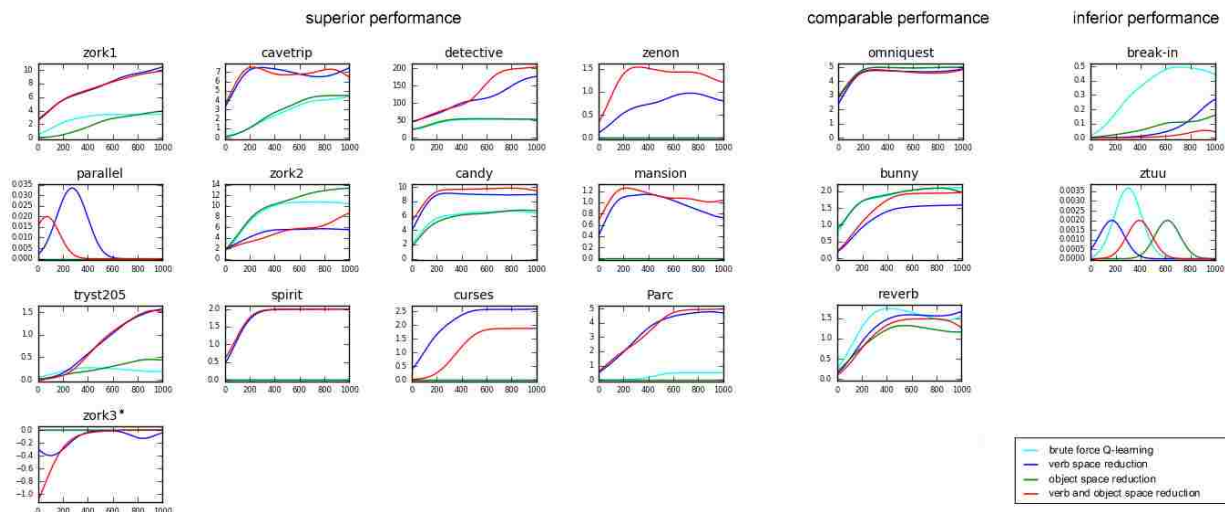


Figure 1.5: Learning trajectories for eighteen Z-machine games. No agent scored any points on the remaining 31 games. Average of 10 data runs. \*In zork3, standard smoothing obscured small, sporadic positive rewards obtained by our reduction algorithms.

Figures 1.5 and 1.6 show the performance of our reduction techniques (blue, green, and red lines) when compared to the baseline (cyan). Affordance-based search space reduction improved overall performance on 13/18 games, and decreased performance on only 2 games.

Examination of the 31 games in which no agent scored points (and which are correspondingly not depicted in Figures 1.5 and 1.6) revealed three prevalent failure modes: (1) The game required prepositional commands such as ‘look at machine’ or ‘give dagger to wizard’, (2) The game provided points only after an unusually complex sequence of events, (3) The game required the user to infer the proper term for manipulable objects. (For example, the game might describe ‘**something** shiny’ at the bottom of a lake, but required the agent to ‘get shiny **object**’.) Our test framework was not designed to address these issues, and hence did not score points on those games. A fourth failure mode (4) might be the absence of a game-critical verb within the 1000-word manipulation list. However, this did not occur in our coarse examination of games that failed.

### 1.5.1 Alternate Reduction Methods

We compared our affordance-based reduction technique with four other approaches that seemed intuitively applicable to the test domain. Results are shown in Figure 1.6.

**Intrinsic rewards:** This approach guides the agent’s exploration of the search space by allotting a small reward each time a new state is attained. The agent is thus encouraged to traverse multiple game locations, enabling possibilities for higher reward. We call these awards intrinsic because they are tied to the agent’s assessment of its progress rather than to external events.

**Random reduction:** When applying search space reductions one must always ask: ‘Did improvements result from my specific choice of reduced space, or would *any* reduction be equally effective?’ We address this question by randomly selecting 30 manipulation verbs to use during each epoch.

**ConceptNet reduction:** In this approach we used ConceptNet’s CapableOf relation to obtain a list of verbs relevant to the current object. We then reduced the agent’s manipulation list to include only words that were also in ConceptNet’s word list (effectively taking the intersection of the two lists).

**Co-occurrence reduction:** In this method, we populated a co-occurrence dictionary prior to training using the 1000 most common verbs and 30,000 most common nouns in Wikipedia. The dictionary tracked the number of times each verb/noun pair occurred within a 9-word radius of each other. During game play, the agent’s manipulation list was reduced to include only words which exceeded a low threshold (co-occurrences  $> 3$ ).

Figure 1.6 shows the performance of these four algorithms, along with a baseline learner using a 1000-word manipulation list, on the sixteen games presented in Figure 1.5. Affordance-based verb selection improved performance in most games, but the other reduction techniques fell prey to a classic danger: they pruned precisely those actions which were essential to obtain reward.

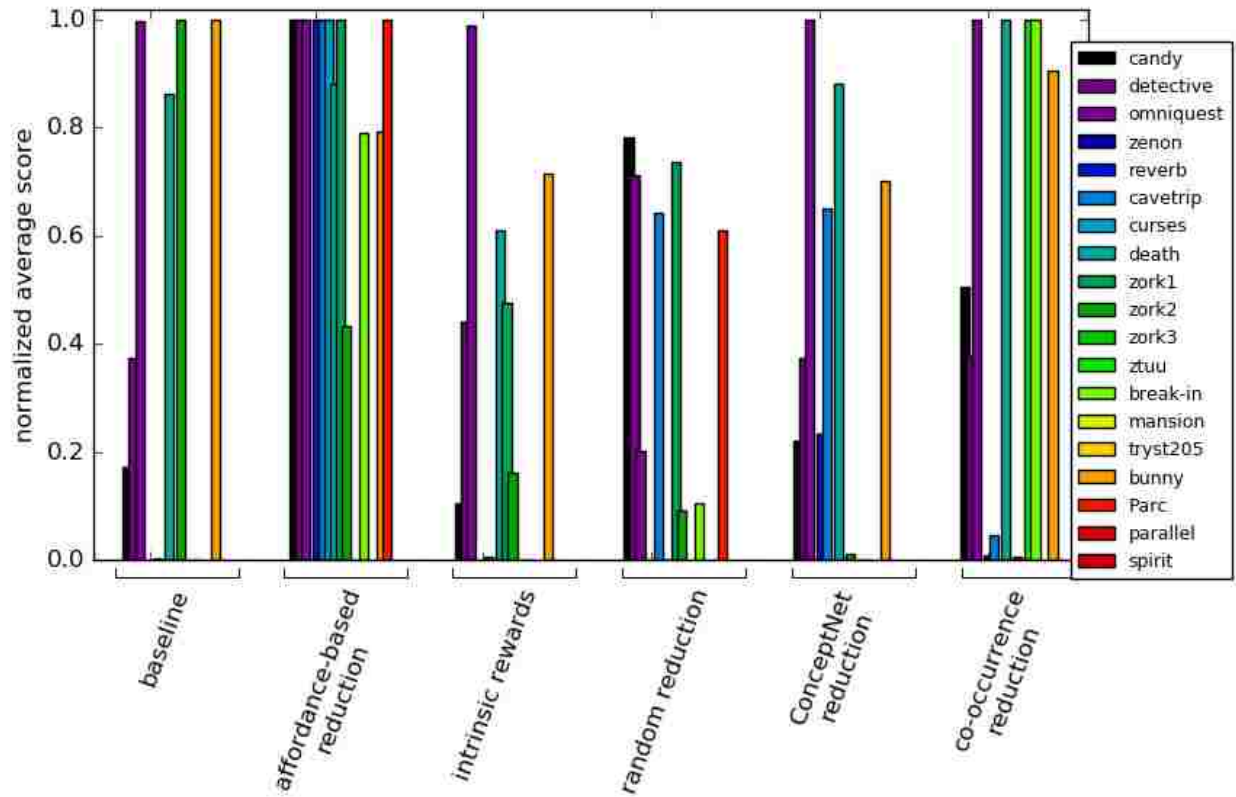


Figure 1.6: Five verb space reduction techniques compared over 100 exploration epochs. Average of 5 data runs. Results were normalized for each game based on the maximum reward achieved by any agent.

Affordant selection	Random selection
decorate glass	continue quantity
open window	break sack
add table	result window
generate quantity	stay table
ring window	build table
weld glass	end house
travel passage	remain quantity
climb staircase	discuss glass
jump table	passage

Figure 1.7: Sample exploration actions produced by a Q-learner with and without affordance detection. The random agent used nouns extracted from game text and a verb list comprising the 200 most common verbs in Wikipedia.

### 1.5.2 Fixed-length Vocabularies vs. Free-form Learning

An interesting question arises from our research. What if, rather than beginning with a 1000-word vocabulary, the agent was free to search the entire English-language verb space?

A traditional learning agent could not do this: the space of possible verbs is too large. However, the Wikipedia knowledge base opens new opportunities. Using the action selection mechanism described in Section 2.4.1, we allowed the agent to construct its own manipulation list for each state (see Section 2.3.1). The top 15 responses were unioned with the agent’s navigation and essential manipulation lists, with actions selected randomly from that set.

A sampling of the agent’s behavior is displayed in Figure 1.7, along with comparable action selections from the baseline agent described in Section 2.4.1. The free-form learner is able to produce actions that seem, not only reasonable, but also rather inventive when considered in the context of the game environment. We believe that further research in this direction may enable the development of one-shot learning for text-based adventure games.

## 1.6 Conclusion

The common sense knowledge implicitly encoded within Wikipedia and other online corpora opens new opportunities for autonomous agents. In this paper we have shown that previously

intractable search spaces can be efficiently navigated when word embeddings are used to identify context-dependent affordances. We have also shown that, in the domain of text-based adventure games, this approach is superior to several other intuitive methods.

We emphasize that, although our initial experiments have been restricted to text-based environments, the underlying principles apply to any domain in which mappings can be formed between words and objects. Steady advances in the fields of object recognition and semantic segmentation, combined with improved precision in robotic systems, suggests that our methods are readily applicable to systems including self-driving cars, domestic robots, and UAVs.

## **Acknowledgements**

Our experiments were run using Autoplay: a learning environment for interactive fiction (<https://github.com/-danielricks/autoplay>). We thank Nvidia, the Center for Unmanned Aircraft Systems, and Analog Devices, Inc. for their generous support.

## Chapter 2

### Harvesting Common-Sense Navigational Knowledge for Robotics from Uncurated Text Corpora

by Nancy Fulda, Nathan Tibbetts, Zachary Brown and David Wingate,  
1st Annual Conference on Robot Learning (JMLR/CoRL 2017).

#### Abstract

As robotic systems are deployed into everyday situations, the need for abstract reasoning becomes more pronounced. The ideal robotic assistant should be able to understand verbal commands and work independently to fulfill human-prescribed goals, even if instructions are ambiguous or circumstances change. This paper presents a new algorithm for high-level reasoning based on Euclidean representations of words and their meanings. Rather than using ontologies or knowledge graphs, we model information about the world as a learned geometry of the contexts in which human beings tend to use each idea. Building on the analogy algorithms utilized by Mikolov et al. [100], we perform mathematical operations on the vector space to infer responses to previously unseen problems, and apply our method to a sequence of semantic reasoning tasks in order to answer questions such as ‘Where can I find a dustpan?’, ‘Where do the crayons belong?’, and ‘What transportation method will bring me to the airport?’. Our Directional Scoring Method (DSM) returns a ranked list of possible responses, many of which are plausible answers to the query. Additionally, DSM’s top-ranked response is significantly more likely to be correct than the top-ranked responses of naive analogy estimations.



## 2.1 Introduction

The field of robotics is transitioning from automation to autonomy. Rather than performing specialized tasks repeatedly, modern systems are expected to behave intelligently in situations where human input is not immediately available and where environmental circumstances cannot be predicted in advance. To excel under these conditions, a robotic system must be equipped with general-purpose knowledge about its environment, the components and prerequisites of potential objectives, and the behavior of other entities. Such knowledge is often represented in the form of ontologies and knowledge graphs [131, 145, 160], but although these structures are easily integrated with robotic reasoning systems, they fail to represent the full complexity of human thought. They also frequently require hand-coding, an expensive and time-consuming process that is susceptible to errors of omission.

In this paper, we take an alternate approach to common-sense reasoning. Following the example of machine learning researchers, we model knowledge about the world as geometric points extracted from uncurated text corpora [9, 97, 116]. *Word embeddings* are trained based on local context, producing a model in which words that tend to appear in similar contexts are proximate to one another. Although these embeddings are trained exclusively based on word co-occurrence, prior work [100] [45] has demonstrated that general purpose knowledge about the world is implicitly encoded in the resulting vector space. For example, it is possible to perform an *analogy query* by providing input of the form  $A:B :: C:D$ , where A, B, and C are given words and D must be inferred. For example, given ‘Madrid:Spain :: Paris:D’, an algorithm should return  $D=France$ , but to do that, the algorithm must ‘know’ in some sense that Paris is associated with France, which is a common-sense fact.

There is a problem, however. When presented with queries that have more than one correct answer, such as ‘microwave:kitchen :: pillow:?’ , the traditional analogy method breaks down (usually by returning a synonym for one of the source words). Perhaps for this reason, word embeddings have seldom been seriously considered as potential knowledge bases for real robots.

This paper presents a novel algorithm for performing analogy queries under conditions which present multiple possible correct answers. The results constitute a valuable proof of concept that word embeddings implicitly encode common-sense facts that are useful to a robot, such as where objects are commonly located and how one might best travel to a given destination. In particular, we show how *canonical vectors* can be used to capture high-level semantic queries, we illustrate that naive extrapolation is inappropriate for the hypersphere topology in which words are typically embedded, and we present an improved methodology for performing semantic queries about everyday objects and environments.

## 2.2 Related Work

In 2013, Mikolov et al. [97] presented word2vec, a learning algorithm that produced vector representations of words based on the ability of a neural network to predict their contexts. They are not the first to perform such a feat, but their machine learning approach was more flexible than the statistical methods [88, 149] that preceded them and more efficient than similar approaches which used more complex network architectures [7, 9, 60]. Perhaps most importantly, Mikolov et al. were able to observe a surprising array of linguistic regularities in the word vectors produced by their training algorithm, including the ability to find gender-paired terms, map countries to the names of their capitols, and identify the present and past tenses of verbs [100].

The result quickly caught the attention of the research community, raising the profile of related algorithms like GLoVE [116] and skip-thought vectors [74]. Multiple approaches have been proposed to improve the quality of the embedding space [82, 98] and to improve algorithm performance on analogical reasoning tasks [55, 83]. In situations where a specific type of word relationship is to be applied repeatedly, the use of a centroid or averaged analogy vector has proven quite effective [15, 45]. Other researchers have focused on algorithms that blend euclidean distance or cosine similarity with directional filters to select desired elements from a vector space. Lukac et al. [94] created a directional filter that minimizes the sum of

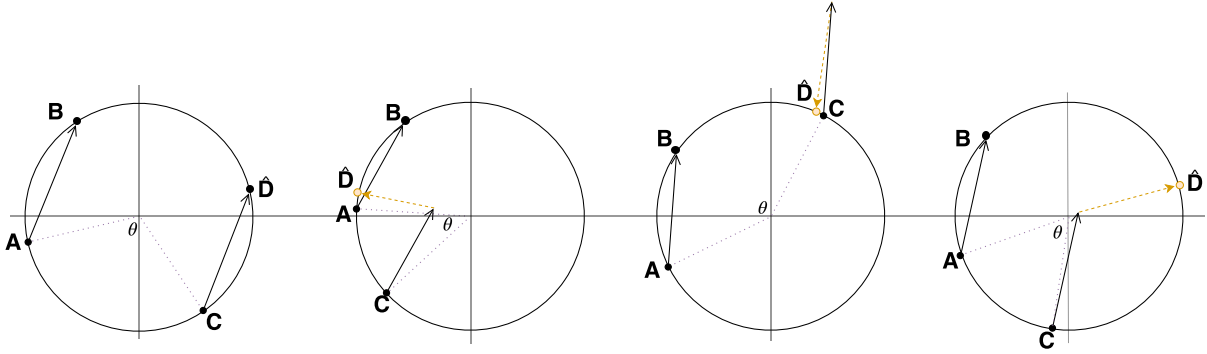


Figure 2.1: Vector operations in a 2-dimensional slice of an  $n$ -dimensional unit hypersphere. The three illustrations on the right represent a possible failure mode. In the middle two illustrations, when analogy vector  $\vec{AB}$  is applied to point  $C$ , the resulting endpoint tends to lie close to one of  $A$ ,  $B$ , or  $C$ . The specific failure depends on the angle  $\theta$  and the length of  $\vec{AB}$ .

weighted angular distances to remove impulses and outliers from an image, while Chen [19] examines a variety of weight functions for spatial autocorrelation.

Recent work in our laboratory applies these concepts at a more abstract level by using word embeddings to automatically detect *affordances* [45], meaning the set of actions that can be performed on a given object. The quality of extracted affordances was evaluated using simulated text-based environments, in which the agent’s maximum score increased by over 75%. In this paper, we build on prior work by improving analogy quality and expanding the types of useful information that can be harvested from unstructured text.

### 2.3 Embeddings and Analogies on Hyperspheres

When discussing Mikolov et al.’s results, enthusiasts often describe analogy relations as a matter of simple vector addition: take the vector for the word ‘king’, subtract from it the vector for ‘man’ and add the vector for ‘woman’, and, *presto!* The result is the vector for ‘queen’. This description is incomplete. In the tagged embedding space used for our experiments, the closest word to the point  $\vec{king} - \vec{man} + \vec{woman}$  is not  $\vec{queen}$ , but instead  $\vec{king}$ . ( $\vec{queen}$  comes in a close second.) This tendency of analogy operations to produce results

close to their origins is well documented and easily verifiable [55, 86]. Take any analogy pair  $A:B::C:D$ , and the vector corresponding to  $\vec{C} + \vec{B} - \vec{A}$  is likely to lie closer to A, B, or C than to any other word in the embedding space.

Mikolov et al. addressed this problem by explicitly excluding all three analogy source words from the analogy results - a straightforward tactic that was introduced in [97] and works remarkably well. However, while this approach yields impressive results on syntactic analogies, it is far less effective on more challenging derivative and lexicographic analogies [55]. We hypothesize that this is due to synonyms and morphological forms of the source words, which would naturally be located nearby, and which would not be eliminated by source word exclusion.

In this paper, we take a different approach by considering the shape of the embedding space: Word2vec analogies are trained using softmax, a normalization function that centers vectors at the origin, thus constraining the final word representations to the surface of a unit hypersphere. Under these conditions, many translations of relational vector  $\vec{r}$  formed using the method  $\vec{r} = \vec{A} - \vec{B}$  will cause the endpoint of  $\vec{r}$  to point away from the surface of the hypersphere. When an algorithm queries for the closest words to the endpoint of  $\vec{r}$  after performing a translation on  $\vec{r}$ , the operation is roughly equivalent to projecting  $\vec{r}$  onto the surface of the hypersphere and then performing the query. Therefore, a sufficiently large angle between  $\vec{A}$  and  $\vec{C}$  will result in a vector whose projection onto the hypersphere lies close to one or more of  $\vec{A}$ ,  $\vec{B}$ , or  $\vec{C}$  (see Figure 2.1).

Accordingly, we present DSM, a directional scoring method that compensates for this tendency to return to the source words of the analogy. Rather than excluding source words explicitly, DSM gives precedence to words that lie along an extension of the canonical analogy vector, thus decreasing the likelihood that source words or their semantic neighbors will be selected as an analogy response.

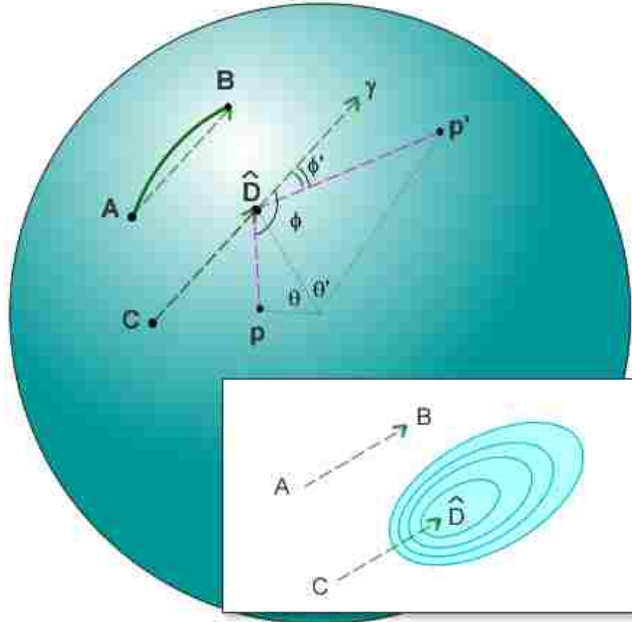


Figure 2.2: DSM calculation. If we take  $\vec{z} = \overrightarrow{AB}$  as a (degenerate) canonical analogy vector, then candidate analogy responses are scored based on the criteria  $S_{dsm} = \alpha D_C(\vec{p} - \hat{D}, \vec{z}) + D_C(\vec{p}, \hat{D}) = \alpha(1 - \cos(\phi)) + (1 - \cos(\theta))$ . Inset: DSM search pattern. Rather than selecting candidate points based solely on proximity, DSM searches in expanding rings that are elongated in the direction of the analogy vector.

## 2.4 Improving Analogies with Canonical Vectors and Directional Scoring

The primary contribution of this work lies in the combination of an averaged relation vector with a directional scoring method (DSM) in order to navigate the embedding space more effectively. Rather than accepting Euclidean proximity as the sole determinant of relevance, we instead evaluate candidate solutions based on a combination of proximity to the the vector offset endpoint and orientation with respect to the analogy vector’s trajectory. We show that DSM matches the performance of traditional offset methods on the Google Analogy Test Set, and that it outperforms them by a factor of 10% to 50% on a more challenging set of analogical reasoning tasks with multiple correct answers.

### 2.4.1 Canonical analogy vectors

As noted by [15, 34, 45], analogy performance can be improved by averaging multiple examples of the type of relationship that is sought. DSM applies this principle as an initial step prior to directional scoring. A set of canonical examples  $A_i, B_i \in V$  is compiled, where  $A_i$  and  $B_i$  are natural language source words and  $V$  is the model’s vocabulary. Let  $\vec{A}_i$  and  $\vec{B}_i$  be the vector representations of  $A_i$  and  $B_i$  and  $\vec{AB}_i = \vec{B}_i - \vec{A}_i$ . The canonical analogy vector  $\vec{z}$  is defined as  $\vec{z} = 1/n \sum_i \vec{AB}_i$  where  $n$  is the number of canonical examples used.

Ideally, the canonical examples should be high-quality exemplars of the relationship sought, and should use words whose meaning is unambiguous and not clouded by multiple possible interpretations. However, as demonstrated in Section 3.5, a group of less stringently selected canonical examples can also function effectively.

### 2.4.2 Scoring Algorithm

DSM uses a weighted sum of (a) cosine similarity and (b) alignment with the analogy vector in order to select candidate analogy solutions. The DSM score  $S_{dsm}$  of a word vector  $\vec{p}$  is calculated as:

$$S_{dsm} = \alpha D_C(\vec{p} - \hat{D}, \vec{z}) + D_C(\vec{p}, \hat{D}) \quad (2.1)$$

where  $D_C$  is the cosine distance between two vectors,  $\vec{z}$  is the canonical analogy vector,  $\hat{D} = \vec{C} + \vec{z}$  is the endpoint of the offset operation,  $\vec{C}$  is the vector representation of the natural language source word to which the offset operation was applied, and  $\alpha$  is a scale factor. Intuitively, this equation can be viewed as attempting to simultaneously minimize both  $\vec{p}$ ’s distance from endpoint  $\hat{D}$  and the divergence of  $\vec{p} - \hat{D}$  from the canonical analogy vector. The algorithm’s preferred response word minimizes DSM score, as shown in in Algorithm 1. If a ranked list of response words is desired, the algorithm orders words by increasing DSM score and returns the top  $k$  candidates.

---

**Algorithm 3** - Directional Scoring Method (DSM) Analogy

---

**Inputs:**

$W$  = Tensor containing all word vectors in the embedding space. Note that all elements of  $W$  are already unit length.  
 $\mathbf{c}$  = Word-vector on which to apply analogy, following the convention  $\mathbf{a}:\mathbf{b} :: \mathbf{c}:\mathbf{d}$ .  
 $\gamma$  = Canonical analogy vector

**Parameters:**

$\alpha$  = Scaling factor. In general practice, alpha=0.3 yields good results.

**Output:**

$\mathbf{d}'$  = Directionally scored analogy response, an element of  $W$ .

```
1:  $\hat{\mathbf{d}} \leftarrow \mathbf{c} + \gamma$  // Analogy response for a standard offset analogy; center of search
2:  $D_C \leftarrow 1 - W \cdot \hat{\mathbf{d}}^T \|\hat{\mathbf{d}}\|$ 
3:  $W' \leftarrow W - \hat{\mathbf{d}}$ 
4: for  $w'_i \in W'$  do
5:    $\mathbf{h}_i \leftarrow \|w'_i\|$  // Building  $\mathbf{h}$ , a list of vector norms
6: end for
7:  $D'_C \leftarrow 1 - W' \cdot \gamma^T \mathbf{h} \|\gamma\|$  // Element-wise division
8:  $D_{dsm} \leftarrow \alpha D'_C + D_C$ 
9:  $m \leftarrow \min(D_{dsm})$ 
10:  $i \leftarrow \text{index of } m \text{ in } D_{dsm}$ 
11: return  $W_i$  // This is our response word,  $\mathbf{d}'$ 
```

---

## 2.5 Quantitative Analysis

To evaluate the DSM algorithm, we compared four analogy variants on two test sets: The Google Analogy Test Set and a new dataset introduced in this paper, the BYU Analogical Reasoning Dataset<sup>1</sup>. Both of these evaluation tasks require the algorithm to infer the correct answer to analogies having the format A:B::C:D. Each task is broken into subtasks based on the type of analogy being evaluated.

### 2.5.1 Algorithms tested

We tested four algorithm variants:

**Offset:** The averaged canonical vector  $\vec{z}$  is added to source word  $\vec{C}$  to obtain point  $\hat{D}$ . Candidate solution words  $p_i$  are returned in order of increasing cosine distance from  $\hat{D}$ .

**Offset with exclusion:** The averaged canonical vector  $\vec{z}$  is added to source word  $\vec{C}$  as above, but the source word C is excluded from consideration when evaluating candidate solution words.

---

<sup>1</sup><https://github.com/NancyFulda/BYU-Analogical-Reasoning-Dataset>

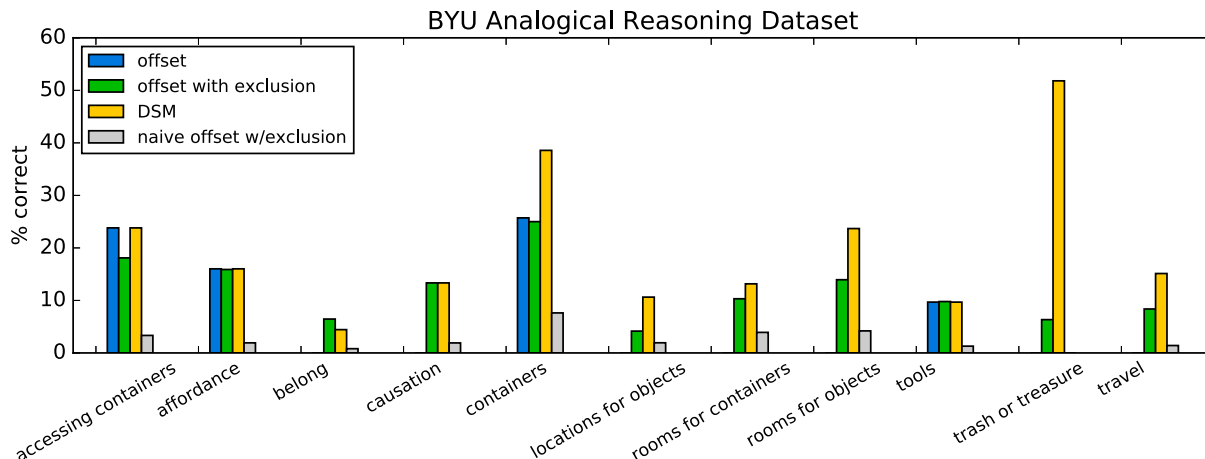


Figure 2.3: Algorithm performance on a sequence of analogical reasoning tasks, showing the % of queries for which the algorithm’s *first* response word was the correct answer. Algorithms utilizing a canonical analogy vector are dramatically superior to the naive application of vector  $\vec{AB}$  to  $\vec{C}$ . Directional scoring improves performance even further.

**DSM:** The averaged canonical vector  $\vec{z}$  is added to source word  $\vec{C}$  and candidate solution words are selected using the scoring method described in Section 3.4.2. No explicit exclusion of source words is applied.

**Naive offset with exclusion:** Rather than using a canonical analogy vector, this method naively applies the vector  $\vec{AB}$  to the source word  $\vec{C}$  to obtain an offset endpoint  $\hat{D}$ . Candidate source words are selected in order of increasing cosine distance from  $\hat{D}$ , with source words A, B, and C excluded from consideration.

The embedding space used for this experiment is identical to that used in [45]. It was trained on a part-of-speech tagged Wikipedia text corpus using Mikolov et al.’s skip-gram method [97]. The final embedding space has 100 vector dimensions and a vocabulary size of approximately 1.5 million words and symbols. Canonical examples for the construction of  $\vec{z}$  were taken from the AB and CD pairs of the first 10 entries of each analogy test set, with duplicates removed. After a coarse parameter search, we set DSM’s  $\alpha$  parameter to 0.3, a value that maximized performance for most (but not all) analogical subsets. We are currently investigating ways to initialize  $\alpha$  programmatically as a function of the canonical examples.



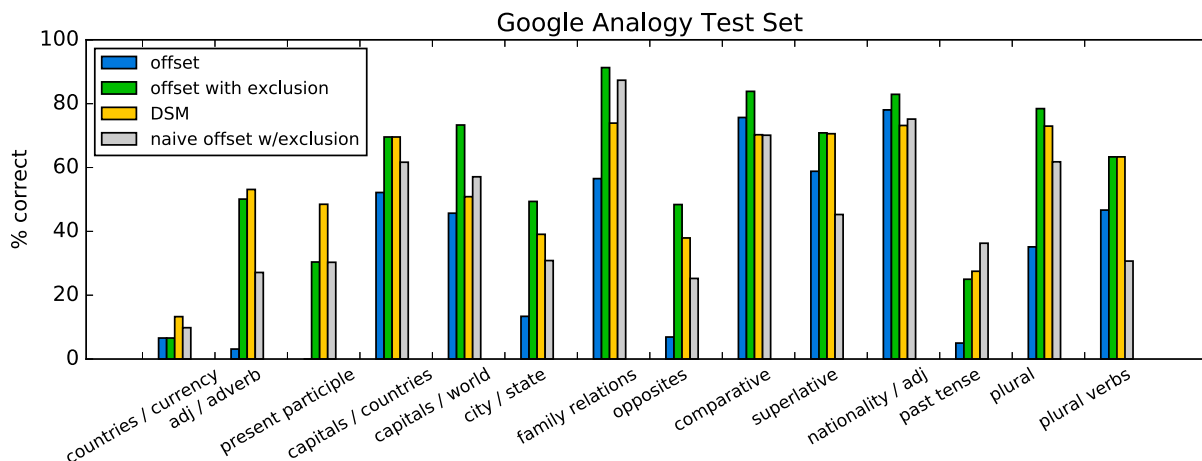


Figure 2.4: Algorithm performance on the Google Analogy Test Set, showing the % of queries for which the algorithm’s *first* response word was the correct answer. On these primarily syntactic datasets, it is more difficult to identify a superior algorithm.

## 2.5.2 Datasets used

The BYU Analogical Reasoning Dataset is a newly created challenge task containing 11,846 analogies of the form A:B::C:D. Subtasks include analogies relevant to robotic navigation and object-based interaction. For example, the ‘Containers’ subtask requires an algorithm to correctly predict that ‘brooms are in closets’, ‘silverware is in a drawer’, ‘milk is in a refrigerator’, and so forth. The ‘Rooms for Objects’ subtask requires predictions about such things as ‘refrigerator is in the kitchen’, ‘potatoes are in the cellar’, and ‘beds are in the bedroom’, while the ‘Tools’ subtask requires knowledge about which objects can be used to enable certain actions, such as ‘cutting requires a knife’, ‘digging requires a shovel’, or ‘baking requires an oven’. These hand-coded analogies are highly abstract and frequently include many-to-one relationships, making them particularly challenging.

The Google analogy corpus was introduced in [97], and is a standard benchmark.

### 2.5.3 Results

Our quantitative results highlight the difference in difficulty between the Google corpus and the BYU Analogical Reasoning Dataset. Whereas the Google dataset consists primarily of clearly-defined relationships with strict one-to-one correspondences<sup>2</sup>, the BYU corpus contains abstract relationships for which multiple answers may seem equally correct. For example, milk can be contained in a jug, but it can also be contained in a bottle. The resulting analogy queries are much more difficult to answer correctly, although as we will see in Section 6, our DSM algorithm is able to return a high percentage of plausible responses.

On the Google dataset, naive offset methods perform almost as well as canonical methods (Fig. 2.4), whereas the more challenging analogical reasoning tasks require a canonical vector  $\vec{z}$  in order to obtain passable results (Fig. 2.3). We are particularly intrigued to note that the performance of our directional scoring method is highly dependent on the specific reasoning task. Although DSM matches or exceeds the performance of traditional offset methods on all subcorpora of the BYU dataset, it sometimes produces improvements of 10% or more. Ongoing research in our laboratory suggests that this may result from distinct differences in the geometry of the source words A,B,C and D, as well as from the clustering behaviors of words within the hypersphere. Further research is required to determine the specific conditions under which DSM produces superior results.

## 2.6 Qualitative Analysis

Consider the thought experiment of a robotic household assistant, designated without loss of generality as ‘RoButler’. To interact effectively with his human controller, RoButler requires common-sense reasoning abilities. A command to ‘tidy up the living room’ presupposes that he understands where common items like sofa pillows or magazines should be placed. When asked to ‘bring me a glass of milk’ he would need to determine that (a) milk is typically in

---

<sup>2</sup>The countries/currency subcorpus is a notable exception, as many countries often share a monetary unit.

the refrigerator, and (b) the refrigerator is typically in the kitchen. This information can then be connected to a planner.

We apply DSM as a tool to facilitate common-sense reasoning and display several sample queries below. Between 10 and 20 canonical examples were hand-selected for each common-sense reasoning task using entries from our Analogical Reasoning Dataset. None of the words in our canonical example set were used as source words for any of the sample queries below.

**Figure 2.5:** Our first query uses canonical vectors extracted from the *travel* dataset, such as:

airport:car :: park:bike    hotel:taxi :: work:train    school:bus :: store:car

In this example, the offset method returns close synonyms of the source word, while DSM proposes candidate words that fit within the category ‘methods of transportation’. Of those, five words (airplane, speedboat, helicopter, jet, boat) represent plausible answers to the original query. Overall, the candidates proposed by DSM are qualitatively superior, a pattern which holds across other queries we’ve tried using this analogy set. Even in cases when offset methods provide a correct first response, most of the other responses fail to fall into the correct word category.

**Figure 2.6:** The next query uses canonical vectors extracted from the *locations for objects* dataset, which encodes knowledge about the locations of common household objects.

milk:refrigerator :: broom:closet    toaster:counter :: book:bookshelf

Traditional methods perform acceptably on this analogy, delivering a valid answer (‘refrigerator’) as the fourth response. Meanwhile, DSM scores a home run by delivering ‘refrigerator’ as its *first* response, followed by a list of plausible locations for alcoholic beverages.

**Figure 2.7:** We now apply a query that uses analogical reasoning to perform a simple classification task: ‘Is this item garbage that should be disposed of, or is it a precious item which must be retained?’

**Query #1: How do I get to Hawaii?**

source word: 'Hawaii\_NN'

OFFSET WITH 'hawaii\_NNS', 'hawaii\_NNP', 'oahu\_NN', 'hawaii\_FW',  
EXCLUSION: 'hawaii\_DT', 'hawaii\_RB', 'honolulu\_NN', 'hilo\_NN',  
'oahu\_NNP', 'hawaii\_ADD', 'maui\_NN', 'kahului\_NN',  
'hilo\_NNP', 'alaska\_NN'

DSM: 'airplane\_NN', 'truck\_NN', 'speedboat\_NN', 'hawaii\_NN',  
'helicopter\_NN', 'oahu\_NN', 'jet\_NN', 'hawaii\_NNS',  
'oahu\_NNP', 'hawaii\_NNP', 'hawaii\_RB', 'kahului\_NN',  
'boat\_NN', 'shuttle\_NN'

Figure 2.5: Query responses calculated using the *travel* dataset.

**Query #2: Where can I get a beer?**

source word: 'beer\_NN'

OFFSET WITH 'coffee\_NN', 'keg\_NN', 'schnapps\_NNS', 'thermos\_NN',  
EXCLUSION: 'refrigerator\_NN', 'bottle\_NN', 'bagel\_NN', 'fridge\_NN',  
'drink\_NN', 'coffeeshop\_NN', 'chocolate\_NN', 'brewed\_VBD',  
'delicatessen\_NN', 'cask\_NN'

DSM: 'refrigerator\_NN', 'fridge\_NN', 'cellar\_NN', 'kitchen\_NN',  
'jacuzzi\_NN', 'pantry\_NN', 'shop\_NN', 'sauna\_NN', 'restau-  
rant\_NN', 'parlor\_NN', 'brewhouse\_NN', 'luncheonette\_NN',  
'thermos\_NN', 'coffeeshop\_NN',

Figure 2.6: Query responses calculated using the *locations for objects* dataset.

wrapper:trash :: cup:treasure peel:trash :: dirt:trash toy:treasure :: sand:trash

We note that similar results hold for other types of queries, such as ‘How do I get to the hospital?’ and ‘Are wood shavings trash?’ In addition, Table 2.1 shows proposed responses for three analogy sets given by DSM vs offset methods using a canonical analogy vector.

Overall, our directional scoring method provides valid responses approximately twice as often as offset methods. Since Table 1 shows that our directional scoring method will quite literally throw the baby out with the bathwater, DSM clearly is not a production-ready system. However, it does provide a valuable proof of concept: Critical common-sense knowledge about

**Query #3: Is jewelry trash?**

source word: 'jewelry\_NN'

OFFSET WITH 'jewellery\_NN', 'jewelery\_NN', 'jewels\_NNS', 'trinkets\_NNS',  
 EXCLUSION: 'jewellery\_NNP', 'priceless\_JJ', 'paraphernalia\_NNS',  
 'valuables\_NNS', 'mementos\_NNS', 'antiques\_NNS',  
 'furniture\_NN', 'souvenirs\_NNS', 'keepsakes\_NNS', 'memorabilia\_NNS'

DSM: 'treasure\_NN', 'treasures\_NNS', 'priceless\_JJ', 'valuables\_NNS', 'jewelry\_NN', 'jewels\_NNS', 'jewellery\_NN',  
 'trinkets\_NNS', 'mementos\_NNS', 'jewelery\_NN', 'paraphernalia\_NNS', 'souvenirs\_NNS', 'memorabilia\_NNS',  
 'antiques\_NNS'

Figure 2.7: Query responses calculated using the *trash vs. treasure* dataset.

a wide variety of topics is implicitly encoded in the vector space. If methodologies can be improved far enough to extract this information reliably, it is potentially superior to hand-coded ontologies or knowledge graphs. Such a system would require virtually no maintenance, would not be prone to errors of omission, could be updated automatically in response to recent news and scientific breakthroughs, and offers possibilities for domain-specific knowledge via the selection of training corpus.

**2.7 Conclusion**

Robotic systems typically rely on ontologies and knowledge graphs for common-sense reasoning. This paper presents an alternative option - word embeddings trained using uncurated text corpora - and demonstrates that a broad spectrum of common-sense knowledge is implicitly encoded within the vector space. This information could potentially be used by a robot that relies on natural language to reason about planning tasks. We have introduced a directional scoring method that nearly doubles rates of correct first responses while simultaneously increasing overall response accuracy. This suggests that with further development of these semantic query algorithms, including adjustments made to compensate for the spherical structure of word embeddings trained using softmax, it may be possible to improve the

Analogy Set	Source Word	Offset + Exclusion	DSM
Travel	hawaii	hawaii* (0)	<b>airplane (5)</b>
	hospital	<b>ambulance (5)</b>	<b>train (8)</b>
	theater	scooter (2)	scooter (4)
	germany	germany* (0)	<b>truck (8)</b>
	mall	<b>car (8)</b>	<b>car (10)</b>
	footbridge	gondola (3)	gondola (3)
	australia	australia* (0)	<b>car (5)</b>
	office	ticket (5)	<b>cab (6)</b>
	moon	moon* (2)	moon (3)
	yosemite	yosemite* (2)	yosemite (4)
Locations for objects	beer	coffee (8)	<b>refrigerator (13)</b>
	bagel	doughnut (5)	cubicle (8)
	knife	noose (1)	noose (3)
	shoe	handbag (0)	drawer (1)
	spoon	<b>tray (2)</b>	bookcase (3)
	remote	nunchuk (1)	remote (0)
	necklace	locket (2)	locket (3)
	marbles	<b>bookcase (5)</b>	<b>bookcase (7)</b>
	backpack	suitcase (0)	suitcase (2)
	bracelet	locket (3)	wallet (3)
Trash or Treasure	jewelry	jewellery (0)	<b>treasure (1)</b>
	pit	pits (1)	pit (1)
	dust	<b>trash (1)</b>	<b>trash (1)</b>
	iphone	iphone* (0)	wii (1)
	shavings	scraps (1)	<b>trash (1)</b>
	apple	sundog (0)	<b>treasure (1)</b>
	baby	mommy (0)	trash (1)
	bathwater	<b>trash (1)</b>	bathwater (1)
	money	loot (0)	money (1)
	dictionary	cyclopaedia (0)	dictionary (1)
Number Correct	-	6 (58)	12 (109)

Table 2.1: Analogy solutions provided by traditional offset methods and DSM when using a canonical averaged analogy vector. The first response provided by each algorithm is shown alongside the number of plausible responses returned (out of a total of 15). Entries are bolded if either the first response was valid or the number of valid responses was greatest. POS tags have been omitted for clarity. \*In these cases, the method returned a differently-tagged version of the source word.

query performance even further. Finally, we have introduced a new Analogical Reasoning Dataset that can be used to benchmark progress in this area, with the hope that other researchers will join us in seeking to create robotic systems that are able to accept high-level commands, adapt these instructions to changing environments, and behave appropriately without micromanagement.

## Chapter 3

### Embedding Grammars

by David Wingate, William Myers, Tyler Etchart, and Nancy Fulda,  
arXiv preprint, 2018

#### Abstract

Classic grammars and regular expressions can be used for a variety of purposes, including parsing, intent detection, and matching. However, the comparisons are performed at a *structural* level, with constituent elements (words or characters) matched exactly. Recent advances in word embeddings show that semantically related words share common features in a vector-space representation, suggesting the possibility of a hybrid grammar and word embedding. In this paper, we blend the structure of standard context-free grammars with the semantic generalization capabilities of word embeddings to create hybrid semantic grammars. These semantic grammars generalize the specific terminals used by the programmer to other words and phrases with related meanings, allowing the construction of compact grammars that match an entire region of the vector space rather than matching specific elements.

#### 3.1 Introduction

Consider the problem of intent detection for natural language processing: to estimate intent, a natural language sentence must be analyzed to determine the underlying goal of the speaker. Ideally, such analysis would be insensitive to the specific words and phrases used, implying the need for a general semantic metric for equivalence.



One way to approach this problem would be to craft a simple rule—say, a regular expression—that describes a set of semantic equivalences. A regex for detecting an intent to deliver a compliment might resemble the following:

```
(i think)? (you're|you are) (beautiful|gorgeous|cute)
```

This would match sentences such as I think you are beautiful, you are cute, and you're gorgeous. However, the ability of this regex to match input sentences is limited by the mental lexicon of the regex designer. What the designer *intended* was to match any phrase that could be construed as a compliment regarding the auditor's general physical appearance, but the produced regex provides only a compact set of examples. The regex does not include all possible synonyms for beautiful, and so it would not, for example, match the sentence I think you are lovely. The power of grammars and regular expressions lie in the compact and efficient representation of combinatorially large sets of sentences, but it is always assumed that terminal matches must be exact. Here, we relax that assumption.

This paper sketches out how to combine the strengths of context-free grammars with the generalization capability of word embeddings to create semantic regular expressions. The grammar provides the coarse overall structure, but individual tokens serve primarily as exemplars that define a region in a word embedding feature space. Any token whose feature representation lies within this region is considered a match. The result is semantic, rather than structural expression evaluation.

### 3.2 Background

Word embeddings [9, 10, 97, 117] represent the meaning of a word as a vector of learned features with interesting linear properties. They have not only been used to evaluate analogies [100][109], but also for document retrieval[15], affordance detection [45], and the generation

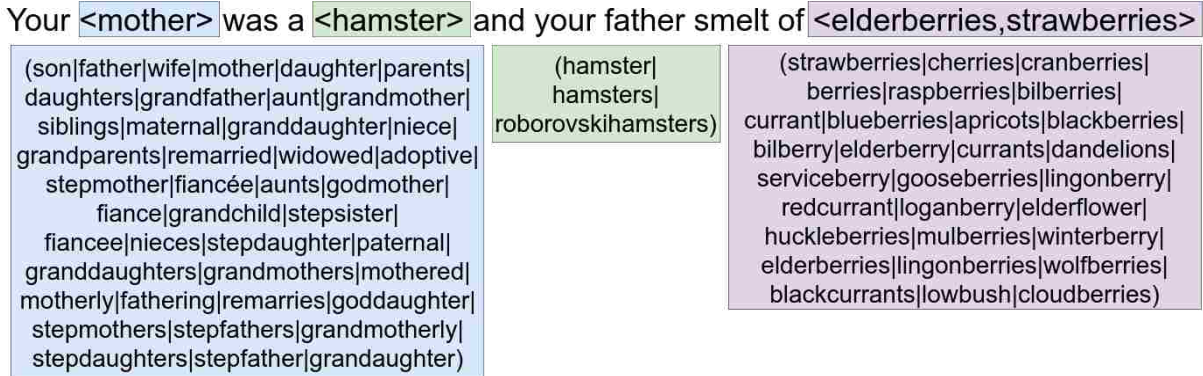


Figure 3.1: Sample regexv expansions obtained using the 500,000 most common tokens in the English-language fastText corpus. Matches were selected using the neighbor expansion method with  $\epsilon=0.35$ .

of sentence-level embeddings [75]. Multimodal embedding algorithms inspired by language models have also been used for text and image alignment [42, 51, 164],

Algorithms for learning word embeddings typically assume that words in similar contexts should have feature representations that are close to one another, implying that semantically related words will be proximately located. Our work utilizes 300-dimensional fastText embeddings trained on a Wikipedia corpus [10] which, like word2vec [97], GloVe [117], and related vector sets, were trained using a neural network tasked with predicting local context. Unlike previous methods, fastText utilizes subword information to speed the training process and generate compositional embeddings for previously unseen words.

Known weaknesses of word embeddings include susceptibility to triangle inequalities and breakdowns in symmetry [109] as well as an inability to represent different semantic meanings of the same word. Researchers are currently exploring disambiguation methods to distinguish between multiple senses of a given word [64, 124] or to combine the strengths of several models [52].

### 3.3 Embedding Grammars

A *context-free grammar* (CFG) is defined as a tuple  $(N, W, R, S)$  where  $N$  is a finite set of nonterminal symbols,  $W$  is a finite set of terminal symbols,  $R$  is a finite set of productions

of the form  $A \rightarrow B$  with  $A \in N$ ,  $B \in (W \cup N)^*$  (where  $*$  is the Kleene star), and where  $S \in N$  is a distinguished start symbol. The language generated by the grammar is the yield of the terminal strings generated with all possible (recursive) substitutions of right-hand side nonterminals with left-hand side nonterminals.

An embedding grammar generalizes classic CFGs: instead of considering terminals  $W$  to be “grounded” in discrete tokens, we instead consider terminals to be grounded in a well-defined region of a feature space. We therefore replace the set of symbols  $W$  with a set of subregions  $W_e$ , where each  $r \in W_e$  is defined as some  $r \subset R^d$ , where  $d$  is the dimension of the feature space.

To connect tokens and the grammar, we require a *word embedding* using one of the methods described in Sec. 3.2 that defines a function  $f$  that maps tokens  $w \in W$  to points in  $R^d$ . We then say that a production  $A \rightarrow B$  is valid if  $B \in W_e$ ,  $w \in W$ , and  $f(w) \in B$ .

This naturally raises the question: how should the subregions be specified? We adopt an exemplar based approach: the grammar designer specifies the subregions indirectly by giving several exemplars, which are then used in conjunction with the word embedding database to automatically generate the subregion.

Figure 3.1 illustrates the idea for a simple BNF grammar given by:

```
S ::= Your REL was a PET and your father smelt of BERRIES
REL ::= mother
PET ::= hamster
BERRIES ::= elderberries
```

where nonterminals are given in all capital letters. Naively, this simple grammar would yield only the single sentence “Your mother was a hamster and your father smelt of elderberries” [54]. However, using embedding grammars, we can interpret the production  $REL ::= elderberries$  as an exemplar, and define the subregion  $r$  as (for example) an epsilon ball centered at  $f(elderberries)$ . This naturally generalizes the yield of the grammar to a wide variety of familial relationships, pets, and berries.

### 3.3.1 Subspace region definition

Subspace definition is independent of the overall grammar structure, and different methods can be used as needed. Here, we outline four possible approaches to subspace definition. (In practical application, we found the neighbor expansion method to be most effective, and use it in our examples throughout the paper.)

**Bounding Box:** A naive subspace can be defined by taking a bounding box of the exemplars. As one might expect, this method does not work particularly well, and it also cannot be applied to productions consisting of only a single exemplar. **Epsilon ball:** In this approach, the centroid of the exemplar set is taken, and all vectors whose cosine distance is less than a predefined value  $\epsilon$  from the centroid are considered to fulfill the production rule. **Covariance subspace:** We define a Mahalanobis distance metric based on the empirical (possibly regularized) covariance of the exemplars. We expected this method to outperform the epsilon ball, since it takes into account the idea that certain components of the vector space have a greater influence on word semantics. In reality, the opposite was true. **Neighbor expansion:** We found this method to be the most successful. In this case, each exemplar in a given production defines a subspace consisting of all vectors whose cosine distance to said exemplar is less than  $\epsilon$ . Thus, a vector which is within the epsilon ball of *any* exemplar is considered to match the production rule. This many-bubbles approach results in close matches for the entire set of exemplars.

## 3.4 A Python Implementation

To test our ideas, we experimented with a simple integration of embeddings and Python's regular expression package (`re`). The Python regex package [155] provides a concise notation for describing sets of character strings, allowing complex string matching based on search patterns entered by a human designer. In Python, `regex` uses a backtracking algorithm rather than translating the regex into an equivalent deterministic finite automaton (DFA) or non-deterministic finite automaton (NFA). We took advantage of `re`'s dynamic nature by

defining and calling a wrapper function which applies regexv before calling each of re’s public API methods.

### 3.5 Experiments

Our experiments used 300-dimensional FastText vectors trained on Wikipedia. A coarse parameter search suggests that  $\epsilon < 0.5$  represents a strong semantic similarity. However, because polar opposites (e.g. beautiful/ugly) appear in highly similar contexts and tend to fall within this range of each other, we adopted a stricter match threshold of  $\epsilon=0.35$ . The neighbor expansion method discussed in Section 3.1 was used for all experiments.

#### 3.5.1 Word extensions

Sample word expansions are shown in Figure 3.1. The common and socially-integral term ‘mother’ (777th most common token) produces a wide range of matches, whereas the less common word ‘hamster’ (33651st most common token) defines a relatively sparse subspace.

Note that grammatically incorrect matches such as ‘widowed’ or ‘motherly’ are unlikely to cause poor system performance, as human-generated text will almost never contain syntactically implausible phrases like “Your widowed was a hamster”. The same holds true for unusual and entertaining words such as ‘roborovskihamsters’. The most problematic error modes involve words that are polar opposites: hot/cold, happy/sad, high/low, because they tend to appear in similar contexts.

Multi-word phrase matching was accomplished by treating each input phrase as the vector average of its individual words, an approach that to our surprise functioned quite well. For example, the vectors for ‘clever’ and ‘man’, when averaged together, produce a vector that is highly similar to the vector for ‘genius’. Further examples are shown in Table 1.

match phrase	exemplar	cos distance
clever man	genius	.3934
clever car	genius	.5776
smart boy	genius	.4822
smart car	genius	.6859
fast car	genius	.7700
smart car	tesla	.5758
smart car	porsche	.4609
a guy who knows everything	genius	.5060
a guy who knows everything	porsche	.7767

Table 3.1: Cosine distance between single-word vectors and compositional vectors created by averaging the elements of a multi-word phrase. The resulting distances correlate with many human estimations of semantic similarity.

<p><b>Regex String:</b> “I would like to call a [taxi]”</p> <p><b>regex:</b> “I would like to call a (taxi car)”</p> <p><i>matches:</i> “I would like to call a taxi”, “I would like to call a car”</p> <p><b>regexv:</b> “I would like to call a &lt;taxi,car&gt;”</p> <p><b>post expansion:</b> “I would like to call a (taxi taxis minibus taxicab taxicabs car cars vehicle driver driving truck automobile suv roadster motorbike racecar minivan motorcar racecars)”</p> <p><i>matches:</i> “I would like to call a minibus”, “I would like to call a taxicab”, etc.</p>
--

Figure 3.2: Comparison between regex and regexv. Here, [taxi] is shorthand for the intent that would include all words that are synonymous with taxi.

### 3.5.2 Intent matching

Figure 3.2 illustrates another example of intent matching. If the string one expects is something like “I would like to call a taxi”, then people could also say “I would like to call a car” and intend the same directive. Typical grammars might define a pattern, “I would like to call a (taxi|car)”, in order to match these semantically equivalent sentences. The problem is that the developer must enumerate all the possible words that could mean the same thing as “taxi” in this pattern. Regexpv solves this issue by allowing a user to define intent words, then automatically expands these to a large list of words that have the same semantic meaning. Figure 3.2 shows how regexpv expands two terms to an entire list, including taxi, minibus, taxicab, car, vehicle, driver, racecar, etc.

## 3.6 Learning and Extensions to PCFGs

The fact that terminals in embedding grammars are defined as regions in a continuous feature space suggest a natural extension to probabilistic context-free grammars (PCFGs). A PCFG is a tuple  $(N, W, R, S, \theta)$ , where  $(N, W, R, S)$  are defined as for a normal CFG, but where each production  $A \rightarrow B$  is annotated with a probability  $\theta_{A \rightarrow B}$ , st  $\sum_{A \rightarrow B \in R_A} \theta_{A \rightarrow B} = 1$ , where  $R_A$  is the subset of productions in  $R$  with left-hand side  $A$ . Informally,  $\theta_{A \rightarrow B}$  is the probability of expanding the nonterminal  $A$  using the rule  $A \rightarrow B$ . This naturally defines a probabilistic generative model, and can be used to (for example) rank parses of ambiguous sentences according to some prior.

Embedding grammars offer an additional generalization: instead of defining matches as hard regions in  $R^d$ , soft matches could be defined using (eg) Gaussians; this suggests fully probabilistic learning algorithms that could take advantage of soft matches and the continuous nature of word embeddings. When combined with, for example, recent probabilistic grammars such as adaptor grammars [66] and fragment grammars [111] that combine Bayesian nonparametrics and PCFGs, this could potentially result in efficient grammar learning

algorithms: learnt grammars could be more compact because of the natural generalization inherent in the word embeddings.

### **3.7 Conclusion**

Word embeddings can be combined with classic grammars to enable semantic matching of regular expressions.

This is advantageous because it reduces human cognitive effort, allowing grammar designers to say what they want more compactly.

Importantly, the matching power of the expression is not limited by the designer's lexicon, but can include words that the designer is not familiar with. Recent advances in word embeddings, including fastText, provide for generalization to previously unseen words based on subword structure.

We have only scratched the surface of the possibilities of embedding grammars. Extensions to PCFGs and the advantages of embeddings vis-a-vis grammar learning are exciting directions for future work.



## Chapter 4

### Informing Action Primitives Through Free-Form Text

by Nancy Fulda, Ben Murdoch, Daniel Ricks and David Wingate,  
NeurIPS Workshop on Visually Grounded Interaction and Language, 2017.

#### Abstract

Autonomous agents require a grounded understanding of language, and multi-modal embeddings seem like a logical way to accomplish this. But once you have a joint embedding space, what is it good for? How do you use it? And most especially, how do you map human input (such as verbal instructions) into a set of behaviors that conform with the human’s desires? This paper presents a method for using distributional sentence representations to prioritize action primitives. Inspired by the human ability to transfer domain knowledge via spoken language, we provide our agent with natural language instructions that describe high-reward behaviors. The agent then uses these instructions to align its actions with human intents. A key advantage of this approach is its flexibility: once a properly-structured embedding space has been obtained, new mappings between instructions and behaviors do not have to be trained using thousands or tens of thousands of labeled examples. Less than twenty data points is sufficient.

#### 4.1 Introduction

This paper presents a method for imbuing agents with high-level domain knowledge via short command phrases given by a human designer. This human guidance is provided at design

time and describes the behaviors a human deems necessary in order to succeed at the current task. However, in order to fulfill these goals the agent must be able to link natural language instructions with action primitives that facilitate them. Previous researchers have proposed using multimodal embedding spaces to map from images to text and back again [25, 31, 123]; however, we argue that simple conversion from one modality to another is usually insufficient to accomplish complex tasks. A multimodal embedding space that maps the word ‘ball’ and an image of a ball to similar locations may enable an agent to convert from natural language text to visual imagery and back again, but how shall the agent choose behaviors that comply with specific instructions such as ‘place the ball on the table’?

We address this question by representing the agent’s sensory input, human guidance, and action primitives in a shared embedding space - in this case, as 4800-dimensional skip-thought vectors. Our current implementation relies entirely on text-based inputs and outputs, but this method is also applicable to embedding spaces that use a joint representation to encode a wide variety of sensor inputs, motor activations, kinematic positions, and object categories. The foundational premise is as follows: sensory input, human guidance, and action primitives are all represented as points in a shared embedding space. Because this embedding space was pre-trained using large numbers of examples, it contains valuable semantic structure. We utilize this structure to leverage a small ( $<20$ ) set of example pairings that map human guidance to action primitives. This mapping, which is essentially a vector, is then used to rank candidate actions based on how well they conform to the provided instructions.

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 accomplishes its task, we require the agent to explore only a small subset of its available action space. This forces the agent to behave in a more focused manner. Interestingly, as the number of allowed actions decreases, the performance of our goal-directed agent tends to improve.

<b>GOAL</b>	<b>VERB</b>	<b>NOUN</b>
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

Figure 4.1: Sample canonical analogy set used find goal-driven actions. The full set in our experiments included 17 goal-verb pairs and 18 goal-object pairs.

## 4.2 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 [76]. In recent years, statistical or count-based models have been replaced by embeddings trained from large amounts of uncurated text. In our work, we utilize the skip-thought embedding model presented by Kiros et al. in 2015 [74], which in turn relies on the word2vec Google News word embeddings trained by Mikolov et al. [96]. Skip-thought vectors [74] 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 [82].) Other related work includes [73], who use an encoder-decoder pipeline to learn multimodal embeddings of both images and text. Key weaknesses of these semantic embedding models include susceptibility to triangle inequalities and breakdowns in symmetry [108] as well as their inability to represent different semantic meanings of the same word. Researchers have proposed training methods to achieve word-sense disambiguation [64, 124] as well as methods for combining the strength of multiple embedding algorithms [52].

Of course, an abstract representation of words in terms of other words has only limited use. In order to forge a connection between sentence representations and grounded behaviors

in the agent’s environment, we build on the affordance-based reasoning methods of Fulda et al. [46]. In some respects, this research resembles the work of Kaplan, Sauer and Sosa in natural language guided gaming [67], 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. Rather than learning a task-specific relationship between input and output, our agent leverages the implicit structure of the joint embedding space to prioritize its actions with respect to a human-defined goal.

### 4.3 Our Algorithm

Following the precedent established in [46], 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. The agent’s objective is to earn points by select action primitives (in this case, a verb and a noun) that (a) function well in combination and (b) progress the agent toward a reward state. Critically, the agent does not incorporate a learning model, nor does it modify its behavior over time. Instead, sensory input, human guidance, and action primitives are jointly represented using a pre-trained skip-thought embedding space [74]. When presented with sensory input, the agent compiles a list of candidate actions which are then filtered based on how well they conform with the human’s guidance.

Our evaluation tasks are drawn from the Autoplay learning environment [125]. These text-based virtual worlds are a challenging and largely unsolved domain: The agent is presented with a text description of its environment such as ‘You are standing in an open field next to a white house.’ The agent extracts nouns from this text (e.g. ‘field’, ‘house’), and must then determine which of the approximately 30,000 verbs in the human language can be paired with the selected nouns in a way that might produce reward. In general, rewards are obtained when the agent either escapes a restricted area, obtains items, or manipulates items.



Figure 4.2: Agent architecture. State descriptions and human guidance, encoded as skip-thought vectors, are used to influence final action selections.

Our key research question is as follows: can an agent designed to pair nouns and verbs in meaningful ways improve its performance by taking human input into account? In other words, is it possible for the agent to ground its language understanding sufficiently to convert free-form text into a prioritization over possible action primitives?

To address this question, we use a small set of example mappings as shown in Fig 4.4. Let  $\{(g_1, v_1) \dots (g_i, v_i)\}$  be a set of data points mapping human utterances to verbs that facilitate the human’s objective (see Fig 4.1, first and second columns) and let  $(g_1, n_1) \dots (g_i, n_i)$  be a set of data points mapping human utterances to objects in a similar fashion (Fig 4.1, first and third columns). Given a state  $s$ , human guidance  $g$ , and a set of game objects  $[o_1, o_2, \dots o_k]$ , our agent prioritizes game objects as follows: we define a canonical guidance-to-noun vector  $G_{noun} = \frac{1}{k} \sum_0^k (n_i - g_i)$ . We now use a dot product to project each candidate object  $o_i$  onto the canonical guidance-to-noun vector  $G_{noun}$  as shown in Figure 4.3. Nouns with the highest dot product are given the most priority.

A canonical goal-to-verb vector is defined similarly:  $G_{verb} = \frac{1}{k} \sum_0^k (v_i - g_i)$ , and verbs relating to each noun are prioritized in the same manner. If more types of action primitives were needed, additional canonical vectors could be similarly created.

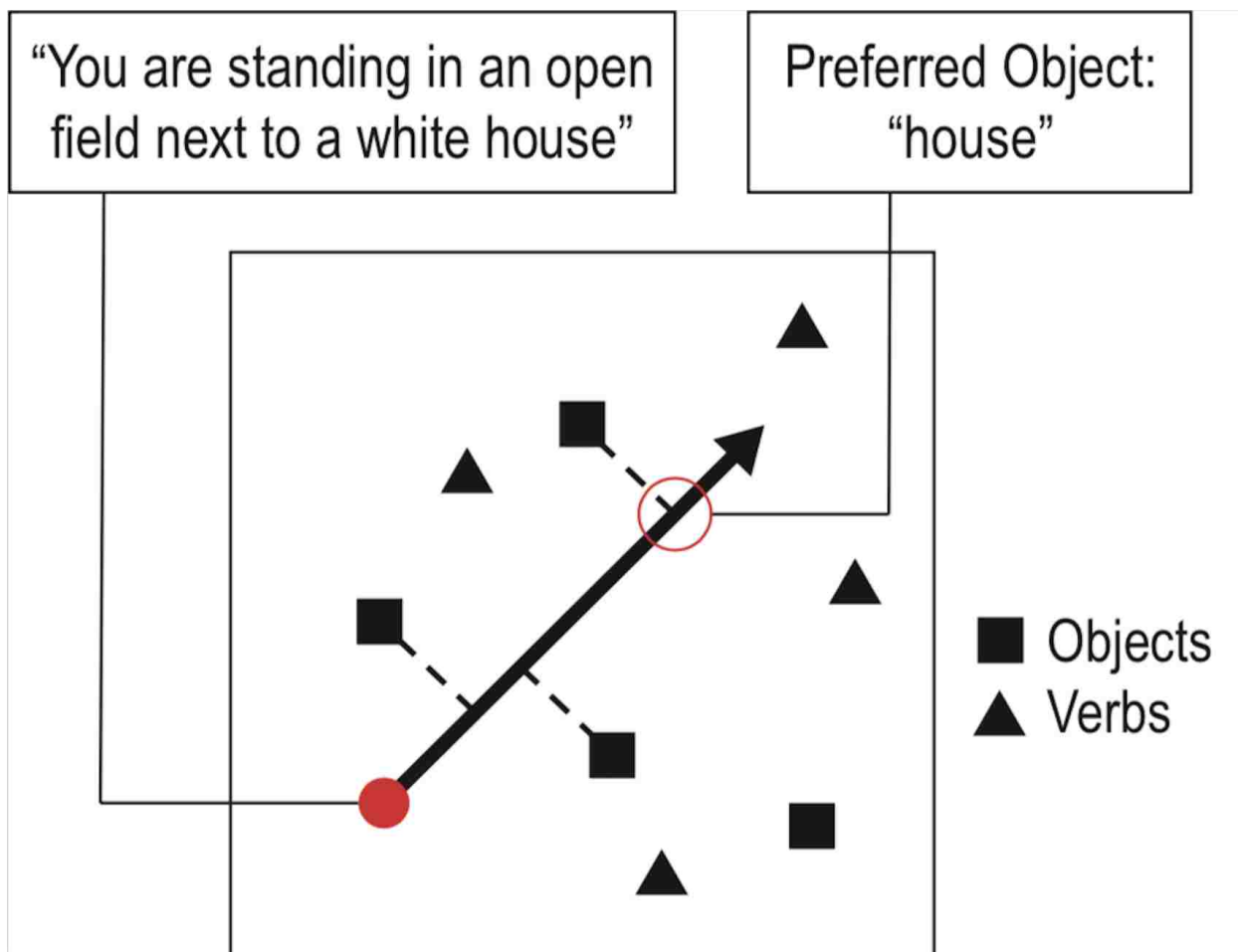


Figure 4.3: Object prioritization based on human guidance. Candidate objects are projected onto the goal vector  $G_{noun}$  and ranked based on magnitude. The system then identifies a set of candidate verbs that are compatible with the chosen object and prioritizes them with respect to  $G_{verb}$ .

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’

Figure 4.4: The human-defined goals for each game, reflecting high-level knowledge of the reward structure. Critically, none of the goal texts produce rewards in their own right; instead, they provide guidance that the agent uses to select promising noun-verb combinations.

#### 4.4 Results

To evaluate our methods, we selected 12 games from the autoplay repository [125]. On each time step, our agent was given one of the human-defined goals depicted in Figure 4.4. 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.

For the most part, these goals were not point-producing actions or even valid action selections within the game. For example, the command ‘enter house’ in *zork1* produces no change of state. In order to achieve this goal, the agent must navigate to the south side of the house and then execute the commands ‘open window’ and ‘enter window’ (which it successfully does). Similarly, 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’.)

Results are shown in Figure 4.5. Three types of agent were compared: a naive agent that randomly selected action primitives, an intelligent agent that paired nouns and verbs

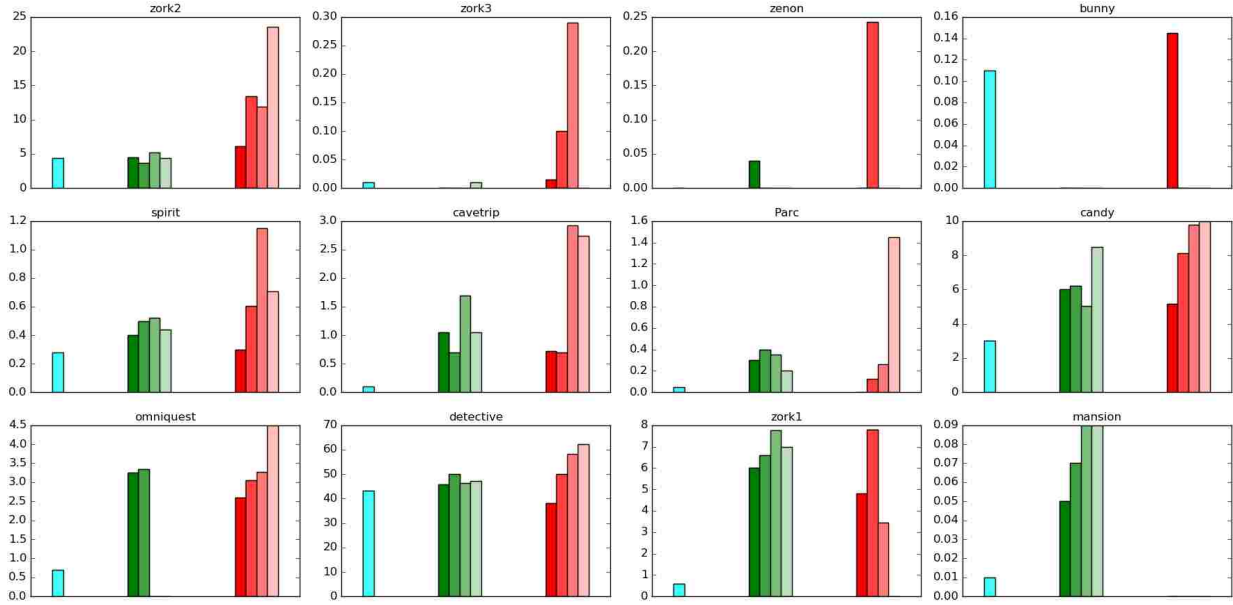


Figure 4.5: **cyan**: random actions. **green**: no human guidance. **red**: human guidance. Clustered bars show changes in game score as the agent is permitted to try [30,15,3,1] verbs for each object. Agents were allowed to interact with each game for 2000 time steps, and the results from 45 data runs were averaged.

using the affordance-based method described by [46], and our agent, which combines the affordance-based pairing method with human guidance delivered via natural language text. Clustered bars indicate the game score achieved by each agent as environmental constraints were increased: on the leftmost bar, the agent was allowed to attempt 30 different verbs for each noun. Moving right, the number of allowed verbs was restricted to 15, 5, and 1 respectively. The number of allowed nouns was similarly restricted, so that the rightmost red bar indicates the agent’s performance when it was allowed to select at most one noun in each state, and was allowed to attempt only its top-priority verb with that noun. Notice that there is a frequent upward trend in the human-guided agent as constraints on the system increase: the agent’s ability to convert human guidance into relevant action primitives enabled it to avoid useless and counterproductive behaviors and generate increased reward in a very small number of steps.



## 4.5 Conclusion and Future Work

Multimodal embeddings offer exciting new potential for grounding agent observations in physical experience, but simply creating a multimodal embedding space is not enough. We must learn how to navigate that space, and how to leverage the multimodal structure to accomplish new tasks without repeating the training process. This paper presents a method for linking high-level domain knowledge in the form of human-generated text with low-level action primitives. Specifically, we use a 4800-dimensional skip-thought embedding space to encode the agent’s inputs, outputs, and guidance offered by a human. In text-based worlds, this approach generates overall improvements in agent performance, and the algorithm is easily generalizable to multimodal embedding spaces. Once a well-structured embedding space has been trained, new prioritizations over behaviors can be created without retraining. A set of twenty or fewer example pairings is sufficient to define the relationship.

In this work we used an embedding space with a single modality (text) to prioritize action primitives. Moving forward we hope to apply our algorithm to multimodal embedding spaces that link text, images, and other sensory input into unified structures that retain semantic meaning. Additionally, these embedding spaces should address the issue of disambiguation between inputs or action patterns that appear in multiple distinct contexts. Stronger multimodal embedding spaces will allow for better inference and average vectors across action modalities. As these embedding spaces become more accurate and reliable, we believe the semantic knowledge found therein will propel autonomous reasoning and natural language understanding to new levels.

## Part III

### Beyond Words and Phrases: Linguistic Reasoning at the Sentence Level

As seen in the previous chapters, experiments with word-level embeddings demonstrate surprising flexibility in the types of information that can be extracted. Using an averaging technique over a small ( $n \leq 15$ ) set of hand-crafted examples, it is possible to extract detailed information: For example, that a horse can be ridden [45], that the best way to travel to Hawaii is by airplane [48], and that a door can be used to enter a house [44].

Sentence-level embeddings are not as simplistic. In order to deal with variable-length input streams, sentence-level embedding algorithms have generally utilized sequence-to-sequence models such as [20, 144], in which a recurrent network is used to encode time series data. These recurrent structures are notoriously difficult to train, often resulting in hidden states that fixate too much on the latter elements of the text. To alleviate this problem, some researchers have moved to an attention-based model such as that used in BERT embeddings [30], or have chosen to formulate sentence embeddings as a simple average of the composite words [5], a method which performs surprisingly well on many tasks, but which omits all subtleties of word order and syntactic structure.

To further complicate matters, the set of grammatically correct English sentences is intractably large, and it is therefore not feasible to perform a nearest-neighbor search to find the sentence that lies closest to the result of an algebraic operation. Instead, a solution must

either be sought from within a set of pre-embedded sample sentences, or a sufficiently robust decoder must be used to map the algebraic result into a grammatically and semantically coherent sentence. In the latter case, a result that is merely in the vicinity of the correct answer is insufficient. We must end up very close to the true solution.

*Chapter 5: Threat Explore, Barter Puzzle: A Semantically-Informed Algorithm for Extracting Interaction Modes* presents an initial foray into the realm of common-sense reasoning at the sentence level. We show that simple mathematical operations within the embedding space are sufficient to solve complex categorization problems far better than a naive baseline, if perhaps not quite as well as we might wish. *Chapter 6: Conversational Scaffolding: An Analogy-Based Approach to Response Prioritization in Open-Domain Dialogs* asks a fundamental and as-yet unanswered question: Is it possible to build a coherent, engaging, and socially appropriate conversational agent without the burden of training a neural dialog model? Using only a set of example conversations and a pre-trained embedding space, we show that it is possible to leverage the analogical structure of the embedding space to select appropriate responses to previously unseen utterances. *Chapter 7: BYU-EVE: Mixed Initiative Dialog via Structured Knowledge Graph Traversal and Conversational Scaffolding* combines the insights from Chapter 6 with the power and flexibility of Amazon’s cloud infrastructure in order to create a dynamic, real-time socialbot capable of interacting with hundreds of users at a time.

Taken together, these chapters paint a compelling picture of the power of common-sense reasoning and the impressive flexibility of sentence-level embedding spaces. However, the question remains: Is it possible to train such embedding spaces to exhibit even more coherence and analogical structure?

## Chapter 5

### **Threat, Explore, Barter, Puzzle: A Semantically-Informed Algorithm for Extracting Interaction Modes**

by Nancy Fulda, Daniel Ricks, Ben Murdoch and David Wingate,  
AAAI Workshop on Knowledge Extraction from Games, 2018.

#### **Abstract**

In the world of online gaming, not all actions are created equal. For example, when a player's character is confronted with a closed door, it would not make much sense to brandish a weapon, apply a healing potion, or attempt to barter. A more reasonable response would be to either open or unlock the door. The term *interaction mode* embodies the idea that many potential actions are neither useful nor applicable in a given situation. This paper presents an algorithm for the Automated Extraction of Game Interaction modes (AEGIM) via a semantic embedding space. AEGIM uses an image captioning system in conjunction with a semantic vector space model to create a gestalt representation of in-game screenshots, thus enabling it to detect the interaction mode evoked by the game.

#### **5.1 Introduction and Related Work**

Video and computer games are a valuable resource for AI researchers. They serve as test domains for novel algorithms [103] [67], provide sensory-rich virtual learning environments [158] and encourage innovation as researchers strive to improve in-game characters [80]. In addition, the emerging field of automated game design learning [113] uses the structure of

the games themselves to extract properties that can be used to (a) improve human play, (b) facilitate virtual character development, (c) create novel tools for developers, and (d) verify that human-specified design properties hold on the model.

In the spirit of this emerging field, we present AEGIM, an algorithm that uses language-based common sense reasoning to distinguish between interaction modes. Game output in the form of pixels or text (or both) is encoded within a 4800-dimensional semantic embedding space trained based on local context<sup>1</sup>. A set of linear classifiers is then used to determine which of several possible interaction modes is evoked by the current situation.

‘Interaction modes’, in the context of this paper, refer to the set of player actions that would be reasonable to execute in the current situation. For example, when confronted with a locked door, it would be reasonable for the player to attempt to unlock it with a key or lockpick. It would be less reasonable to brandish a sword, apply a healing potion, or attempt to barter with the closed door. Similarly, when confronted with an aggressive enemy, one would expect the player to either attack or flee. Under those circumstances, it would not make sense to engage in casual conversation or examine an ornate rug on the floor.

In other words, not all actions are equally preferable in every situation.

Given that the relative value of an action is context-dependent, we wish to identify a set of contexts (i.e. ‘interaction modes’) in which certain actions best apply. In theory, a system capable of identifying such modes could map a game based on behavioral contexts rather than on world geography, a potentially useful diagnostic tool.

Our research utilizes recent work in computer vision [147] [21] to convert pixels into text descriptions of a scene, but goes beyond simple object recognition [78] or semantic segmentation [91] in order to acquire a common-sense representation of the observed items. To do this, we use the skip-thought embedding space [75]. Related semantic embedding spaces include word vectors [97] [116], sentence embeddings using a simple or weighted average of

---

<sup>1</sup>In other words, the vector representation of each sentence is influenced by the sentences that tend to appear before or after it in the input corpus. Word-level semantic spaces trained in this way have been shown to encode common-sense knowledge about the physical and sociological properties of our world [100] [109].

word vectors [38] [29], fixed-length and LSTM-based sentence embeddings [128] [32], and document-level embeddings [81].

This work touches tangentially on the field of affordance detection<sup>2</sup> [173] [139] [107], particularly as explored by [45] in their work with text-based adventure games. Fulda et al. used analogical operations performed in vector space to detect the behaviors afforded by a specific game item. Our work takes this idea one step farther by detecting the current interaction mode of the game (and by extension the subset of actions afforded by the situation) rather than the affordances of a single object.

## 5.2 Methodology

Our algorithm for the Automated Extraction of Game Interaction Modes (AEGIM) begins with screenshots extracted during game play, which are then converted to a plain text representation using an online caption generation tool. We do not require the caption generator to provide a coherent sentence; a simple string of objects or adjectives is sufficient for our purposes.

Once we have a plain text description of an image, we encode the description as a geometric point in a semantic embedding space. For this step, we rely on the skip-thought embedding model [75], which is trained by requiring a neural network to predict both the sentence that directly precedes and the sentence that directly follows each sentence in the training corpus. This is, of course, a mostly futile task: The number of possible antecedents and successors for any given sentence is enormous. However, during the training process the network creates an internal representation that roughly corresponds to each sentence’s semantic meaning. It is this internal representation - the semantic meaning of a sentence - that interests us. We use it to encode each input string as a 4800-dimensional vector that represents the location of the text in skip-thought space.

---

<sup>2</sup>*Affordances* [53] refer to the set of actions that are made possible by an object or situation. For example, a ladder affords the possibility of climbing, while level terrain affords the possibility of running. Gibson asserts that an affordance is neither a property of the environment nor of the actor, but of the complementarity of the two.

<b>Interaction Mode: Threat</b>
‘You see a soldier holding a sword’
‘You are badly wounded’
‘A massive troll bars the path’
‘The bull paws the ground, then charges toward you’
‘The poisonous spider advances, ready with its deadly bite’
‘You are in danger’
‘If you fall from this height, you will die’
‘The battle rages around you’
‘The angry man begins to attack you’
‘You are plummeting to your death, with only a few seconds before you hit the ground’

Figure 5.1: Example texts used to define the ‘Threat’ mode, meaning that an immediate physical danger is present.

Conversion into a semantic embedding space is essential in order to create a gestalt representation of the items in a scene: A hammer alone does not indicate an imminent physical threat, nor does an angry man. But an angry man holding a hammer is cause for immediate concern.

Next, the AEGIM algorithm determines which of several possible interaction modes are indicated by the original screenshot. This is done by comparing the skip-thought vector that represents the image description with the vector embeddings of a set of hand-coded example texts that exemplify each mode. To determine whether an image evokes the interaction mode of ‘Threat’, for example, we would enter sentences like those shown in Fig. 5.1.

We note with interest that these example sentences do not necessarily have to be hand-coded. It is not difficult to imagine scenarios where example text is extracted algorithmically by observing which subset of actions produces reasonable results under specific game conditions.

Once the example texts have been provided, each text is encoded as a skip-thought embedding and an average representative vector is calculated for each example set. New sentences are then classified based on their proximity to these cluster centers. More formally, let  $M = m_1, \dots, m_k$  be the set of interaction modes to be detected and let  $S_i = \{s_i^1, \dots, s_i^n\}$  be the example strings associated with the  $i$ th interaction mode (see Fig. 5.1). Then  $V_i =$

$\{v_i^1, \dots, v_i^n\}$  is the set of 4800-dimensional vector encodings of  $S$  in the skip-thought embedding space, where  $k$  is the number of interaction modes to be classified and  $n$  is the number of examples used to define each interaction mode. For each incoming text  $t$  to be classified, AEGIM determines which interaction modes apply according to the following algorithm:

```

1:  $v_t =$  the vector encoding of text  $t$ 
2: for  $i$  in  $1 \dots k$  do
3:    $d_i = v_t - 1/n \sum_0^n v_i^n$ 
4:    $d_q = v_t - 1/(k-1) \sum_q 1/n \sum_0^n v_q^n, q \neq i$ 
5:   if  $|d_i| < |d_q|$  then
6:     return True
7:   else
8:     return False
9:   end if
10: end for

```

In other words, AEGIM returns *True* for the  $i$ th interaction mode if and only if  $v_t$  is closer to the  $i$ th cluster center than it is to the average of the cluster centers of all other interaction modes. Note that it is possible for AEGIM to output more than one interaction mode for a given image, or no interaction modes at all.

Because skip-thought representations encode the *meaning* of a sentence rather than the *text* or *syntax* of the sentence, this simple linear classifier is sufficient to detect the correct interaction mode in many instances. Accordingly, one could imagine that a more sophisticated classifier might improve performance beyond that reported in this paper.

### 5.3 Results

As a proof of concept for this idea, we collected images from the popular Bethesda game Skyrim (Bethesda Softworks LLC 2013). The dataset consists of 65 images collected during



game play, including peaceful encounters with shopkeepers, wandering animals, and local architecture as well as high-risk encounters with monsters and magic-wielders<sup>3</sup>. Images were then passed through one of three text-generators:

1. **Clarifai** [21], an online object recognition system that returns a list of identified elements within a scene
2. **CaptionBot** [147], an online caption generator that creates a complete sentence describing the contents of an image
3. **Human-generated** captions

The human-generated captions were provided by an 11-year-old girl who has never played Skyrim, is not familiar with our algorithm, and was unaware of how the captions would be used. She was instructed only to ‘give a one-sentence description of what’s in the picture’. Her description was used exactly as given, without prompting, hints, or post-processing. The automatically-generated captions were post-processed as follows: Clarifai returns a list of 20 identified elements for each scene, along with their estimated likelihood. In our experiments, we concatenated the 10 elements with the highest likelihoods into a single text string which was then encoded as a skip-thought vector. CaptionBot prefaces each sentence with a qualifier indicating its certainty about the given description, and sometimes appends a description of whether people in the image appear happy or sad. This supplementary information was omitted; only the core image description was passed through the skip-thought encoder.

The purpose of including human-generated captions was to test the soundness of our algorithm when provided with high-quality text descriptions. This was motivated by the observation that online caption generation systems, which are optimized for photorealistic images of real world objects, perform poorly when presented with game images.

We focused on four interaction modes for this experiment: **Threat** (as depicted in Fig. 5.1); **Explore**, indicating an opportunity to traverse the landscape and discover items

---

<sup>3</sup>No player-controlled characters were harmed during the collection of this dataset.

	<b>AEGIM</b>	<b>keyword baseline</b>	<b>constant</b>
Clarifai	72.3%	8.6%	74.2%
CaptionBot	73.3%	13.3%	74.2%

	<b>AEGIM</b>	<b>keyword baseline</b>	<b>constant</b>
Clarifai	21.9%	14%	43.8%
CaptionBot	31.3%	18.8%	43.8%

Figure 5.2: Top: Classification accuracy, counting each image/category pair individually. Bottom: Exact matches, meaning the percentage of images where all four categories were classified correctly. Human captions were not available for the full dataset, and so are not shown here.

of interest; **Barter**, indicating an opportunity for exchange of goods; and **Puzzle**, indicating that a manual manipulation task is required or available. The values of  $k$  and  $n$  were set to 4 and 10 respectively.

To create reasonable baseline comparisons, we selected two naive classifiers. The keyword method compiles a list of keywords extracted from the AEGIM example texts provided for each interaction mode. During naive classification, an interaction mode was marked as True if any of the keywords appeared within the text to be classified. The constant baseline simply returned the most common classification (‘explore’) in all cases. Results are shown in Figure 5.2. Interestingly, the AEGIM algorithm was able to approach or exceed both baselines with only 10 example texts per category and no online training.

Figures 5.3-5.6 show screenshots from the game along with interaction modes identified by AEGIM. Correct classifications are highlighted using boldface text. Overall (and unsurprisingly), the human-generated captions produce far better results than automated captions. This suggests that it would be worthwhile to either train a caption-generation system on images that more directly align with the task, or to use in-game annotations as an alternative to caption generation.



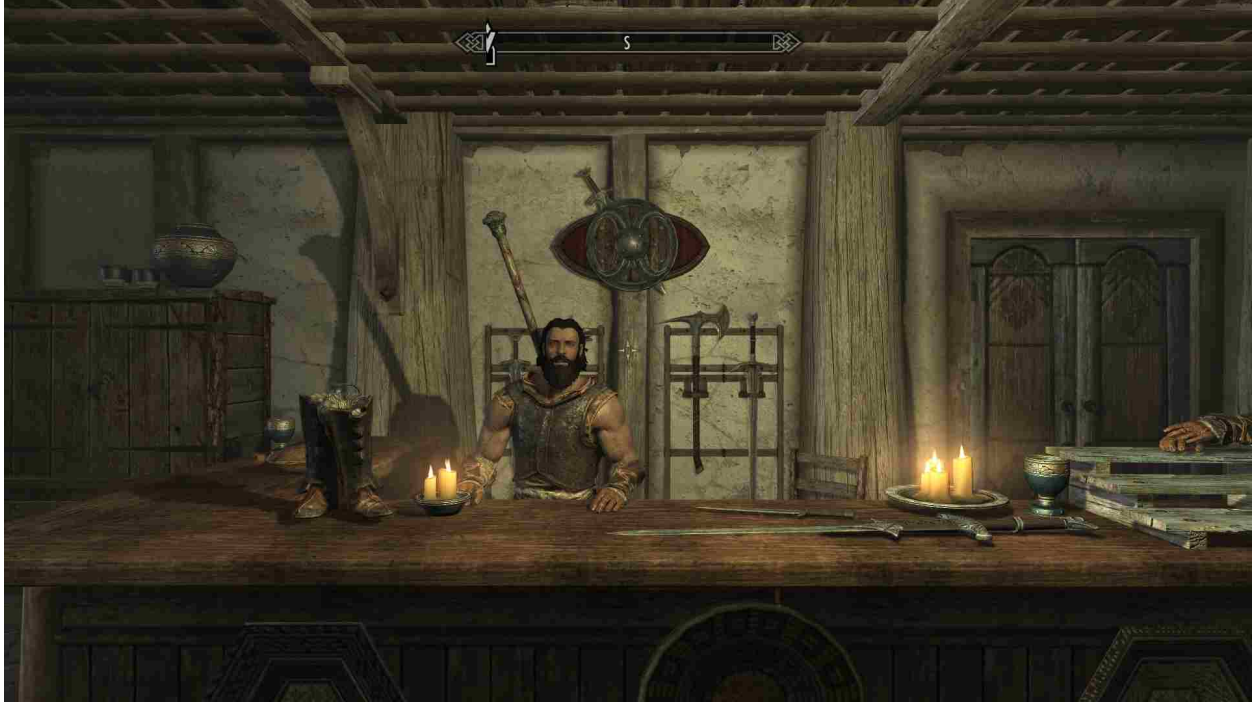
	Generated Text	AEGIM Output
Clarifai:	'no person', 'travel', 'landscape', 'outdoors', 'snow', 'sky', 'mountain', 'daylight', 'winter', 'water'	<b>Explore</b>
CaptionBot:	'A tower with a mountain in the background'	<b>Explore</b>
Human text:	'A windmill near rocky mountains'	<b>Explore</b>

Figure 5.3: Overall AEGIM correctly identifies exploration scenes regardless of captioning method, perhaps because Clarifai and CaptionBot both detect landscape elements like hills, clouds, and buildings.



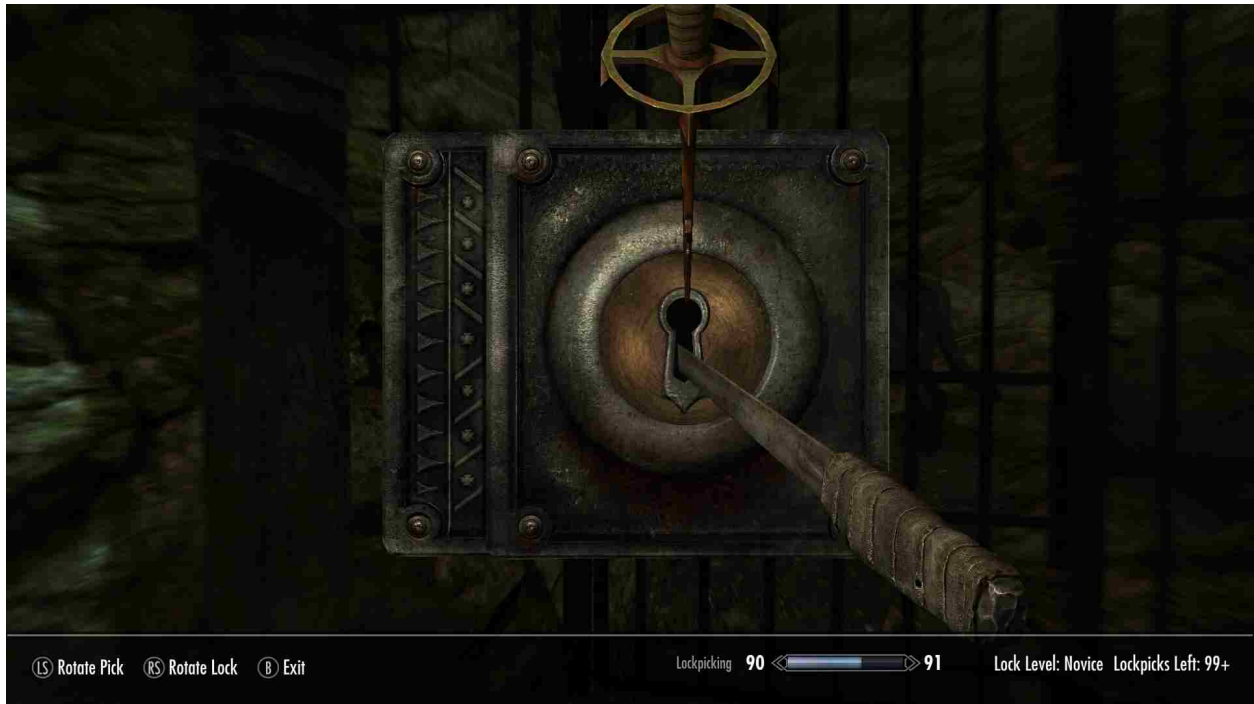
	Generated Text	AEGIM Output
Clarifai:	'people', 'adult', 'smoke', 'flame', 'vehicle', 'military', 'one', 'man', 'weapon', 'war'	Barter
CaptionBot:	'A man jumping over a fire'	<b>Explore</b>
Human text:	'An archer ready to fight against the enemy'	<b>Threat, Explore</b>

Figure 5.4: This is one of the few combat images that was even moderately well-captioned by the automated systems. (Scenes with similar elements were described as 'A group of people jumping' or 'a young man practicing his tricks on his skateboard')



	Generated Text	AEGIM Output
Clarifai:	‘people’, ‘adult’, ‘religion’, ‘indoors’, ‘man’, ‘group’, ‘home’, ‘one’, ‘no person’, ‘travel’	<b>Barter</b>
CaptionBot:	‘A kitchen with wooden cabinets and a fireplace’	-
Human text:	‘A blacksmith that is sitting in his shop, but he also looks very buff’	-

Figure 5.5: In this case the human captioner’s commentary on the shopkeeper’s physique obscured the correct classification. AEGIM correctly returns the interaction mode ‘Barter’ when given the input text ‘A blacksmith sitting in his shop’.



	Generated Text	AEGIM Output
Clarifai:	'rusty', 'old', 'iron', 'security', 'dirty', 'safety', 'no person', 'steel', 'lock', 'door'	<b>Puzzle</b>
CaptionBot:	'A clock that is looking at the camera'	-
Human text:	'A door lock which is trying to be opened with a floating knife and a sharp thingie'	<b>Puzzle</b>

Figure 5.6: It is not difficult to see why CaptionBot mistook the image for a clock. Less obvious is why AEGIM does not consider the clock a puzzle item to be interacted with.

## 5.4 Conclusions and Future Work

AEGIM is a novel and potentially powerful tool for identifying the interaction modes evoked by an image. Preliminary experiments show that it is able to produce correct classifications in a variety of situations.

AEGIM offers the following key advantages over methods that predict interaction modes directly from images: (1) AEGIM’s semantic embedding space can be customized via the selection of input corpus: Skip-thought embeddings trained using game manuals, genre-relevant articles, or in-game dialogue may prove to be particularly effective. (2) By encoding images into the skip-thought embedding space, AEGIM is able to interpret the gestalt meaning of items in a scene rather than evaluating each element independently. (3) AEGIM offers the possibility of dynamically-generated example sets extracted from game-internal images. (4) With AEGIM, image descriptions may be augmented using character dialogue, in-game annotations or other raw text produced by the game engine.

AEGIM’s greatest current weakness lies in the poor quality of the automatically generated captions. More research is required to determine which (if any) of the currently available open-source vision systems is able to produce acceptable descriptions of fictional scenarios. For many games, this limitation may be circumvented by utilizing character dialogue and/or in-game annotations instead of a vision system.

Future work in this area should include the use of neural networks or K-nearest-neighbor classifiers in lieu of linear classification. AEGIM’s performance thus far indicates that the skip-thought space is well-structured for the task of distinguishing between interaction modes; however, the current system is easily foiled by modifying clauses. ‘A dragon flying’ is consistently classified as a threat, but ‘a dragon flying through a cloudy sky’ is not. We anticipate that the use of more sophisticated classifiers will rectify this problem.

Lastly, attention should be given to the task of training high-quality image recognition systems for computer-generated images extracted during gameplay. Such a system would not only be useful for AEGIM, but also for many other potential applications.

## 5.5 Acknowledgements

We thank Alex, Aubrey, and Janika F. for data generation. We thank Nvidia, the Center for Unmanned Aircraft Systems, and Analog Devices, Inc. for their generous support.



## Chapter 6

### Conversational Scaffolding: An Analogy-Based Approach to Response Prioritization in Open-Domain Dialogs

by Nancy Fulda, Tyler Etchart, William Myers and David Wingate,  
to be submitted

#### Abstract

We present *Conversational Scaffolding*, a response-prioritization technique that capitalizes on the structural properties of existing linguistic embedding spaces. Vector offset operations within the embedding space are used to identify an ‘ideal’ response for each set of inputs. Candidate utterances are scored based on their distance from this ideal response, and the top-scoring candidate is selected as conversational output. We apply our method in an open-domain dialog setting and show that the most effective analogy-based strategies outperform both an Approximate Nearest-Neighbor approach and a neural network trained to predict the next utterance in the sequence. We also demonstrate the method’s ability to retrieve relevant dialog responses from a repository containing 19,665 random sentences.

#### 6.1 Introduction

Language is combinatorial in nature, and thus resistant to brute-force information retrieval approaches. The *patterns* of language, however, are far more tractable. In this paper, we use a conversational scaffold corpus to provide examples of these patterns, then use the analogical

structure of a pre-trained embedding space in order to generalize to input sequences that do not closely match any of the scaffold examples.

The basic concept is simple. We begin by encoding a reference corpus, called our *scaffold*, using Google’s Universal Sentence Encoder Lite [17]. Incoming utterances are matched against the scaffold corpus using one of several possible contextualization techniques, and the top  $n$  responses are then used to calculate an analogically coherent response, or *target point*, for each sequence of inputs. This target point does not necessarily represent an optimal reply; it may in fact be the vector embedding of a nonsense sentence. Rather, it functions as a heuristic to gauge whether a specific candidate response is conversationally appropriate.

We examine the effectiveness of our Conversational Scaffolding algorithms on a response prediction task and show that the most effective combinations outperform both an Approximate Nearest Neighbor classifier and a neural retrieval approach. We also contribute an empirical study showing how scaffolding algorithms can be combined with contextual alignment techniques in large, curated datasets in order to improve overall accuracy. Finally, we apply our scaffolding technique in a real-time conversational scenario and show that it is able to retrieve a series of plausible responses to human-generated input.

Our approach has two key advantages: First, it leverages the analogical properties of pre-trained sentence embeddings to identify appropriate responses even when the disparity between the input utterance and the nearest curated example is quite large. Second, it operates directly on a curated reference corpus that can be updated dynamically, without needing to retrain the neural models. Although not directly explored in this paper, we believe this technique has potential for use in conversational systems that extend beyond information retrieval. For example, it might lend itself well as a loss function when training generative models, or when evaluating response quality in iterative self-play scenarios.

## 6.2 Related Work

### 6.2.1 Utterance Retrieval

Retrieval systems for conversational AI have historically relied on statistical models such as Term Frequency-Inverse Document Frequency (TF-IDF) [18, 50, 130] or token-level vector space models [6, 18, 36] with cosine distance used to score candidate utterances. More recently, a variety of neural models for information retrieval have been explored, including a paraphrase matching algorithm utilizing recursive auto-encoders [110], models based on LSTMs and other recurrent units [93, 110], and sequential matching networks that use an RNN to accumulate vectors representing the relationship between each response and the utterances in the context [171].

The recent availability of general purpose pre-trained linguistic embedding spaces such as skip-thought vectors [74], quick-thought vectors [89], InferSent [23], and Google’s Universal Sentence Encoder [17], as well as task-specific embeddings extracted from popular generative models such as HRED [134], present new and interesting possibilities. Bartl and Spanakis [8] use conversational contexts extracted from HRED embeddings to retrieve a set of  $n$  candidate sentences via an Approximate Nearest Neighbor (ANN) algorithm. Each candidate is then scored based on its cosine similarity to each of the other candidates in the retrieved set.

### 6.2.2 The analogical structure of embedding spaces

Our research builds on previous work by leveraging the analogical structure inherent in language, and hence also present in linguistic embedding spaces, to improve response selection. Researchers have shown that pre-trained word embeddings can be used to solve linguistic analogies of the form  $a:b::c:d$  ( $a$  is to  $b$  as  $c$  is to  $d$ ). This is generally accomplished using vector offsets such as [ $king - man + woman \approx queen$ ] or [ $walking - walked + swimming \approx swam$ ] [56, 100, 116]. Query accuracy can be further improved by averaging multiple vector offsets [35, 45] or by extending the length of the offset vector [47].

Our research contribution is the application of vector offset methods to response retrieval in open-domain dialog settings. Rather than evaluating candidate responses based on their strict distance to elements in the reference corpus, we instead rely on the relative distance between pairs of sentences in order to localize an ‘ideal’ response vector, which corresponds to point  $d$  in the classic  $a:b::c:d$  analogical form. Candidates are scored based on their distance from this idealized point.

### 6.3 Conversational Scaffolding

The technique of Conversational Scaffolding, first introduced as an experimental sub-component of [citation omitted for anonymity], is refined in this work by introducing multiple localization techniques and by performing a structured evaluation of response accuracy across a variety of scoring algorithms. We also provide an example of the algorithm in action, showing its ability to retrieve relevant responses from a set of ca. 20,000 randomly-selected sentences.

Figure 6.1 gives an overview of the scaffolding technique. Given a dialog context of variable length, our method first locates a set of high-quality contextual matches within the scaffold corpus. These contextual matches, along with the utterance directly following each context match, are then passed to one of several scoring algorithms. All utterances are encoded using Google’s Universal Sentence Encoder Lite [17], a lightweight but impressively robust embedding model.

#### 6.3.1 Contextual alignment

*Contextual alignment* refers to the process of matching incoming utterances against similar utterance patterns within a scaffold corpus. This can be done naively by using an Approximate Nearest Neighbor<sup>1</sup> algorithm based on a simple Euclidean distance metric<sup>2</sup>. In this paradigm,

---

<sup>1</sup>Approximate approaches, rather than a more rigorous K-Nearest Neighbor algorithm, are used in order to improve computation speed.

<sup>2</sup>Any valid distance metric, such as cosine distance, can be used. We tested both cosine and Euclidean distance, but found the latter to be empirically better.

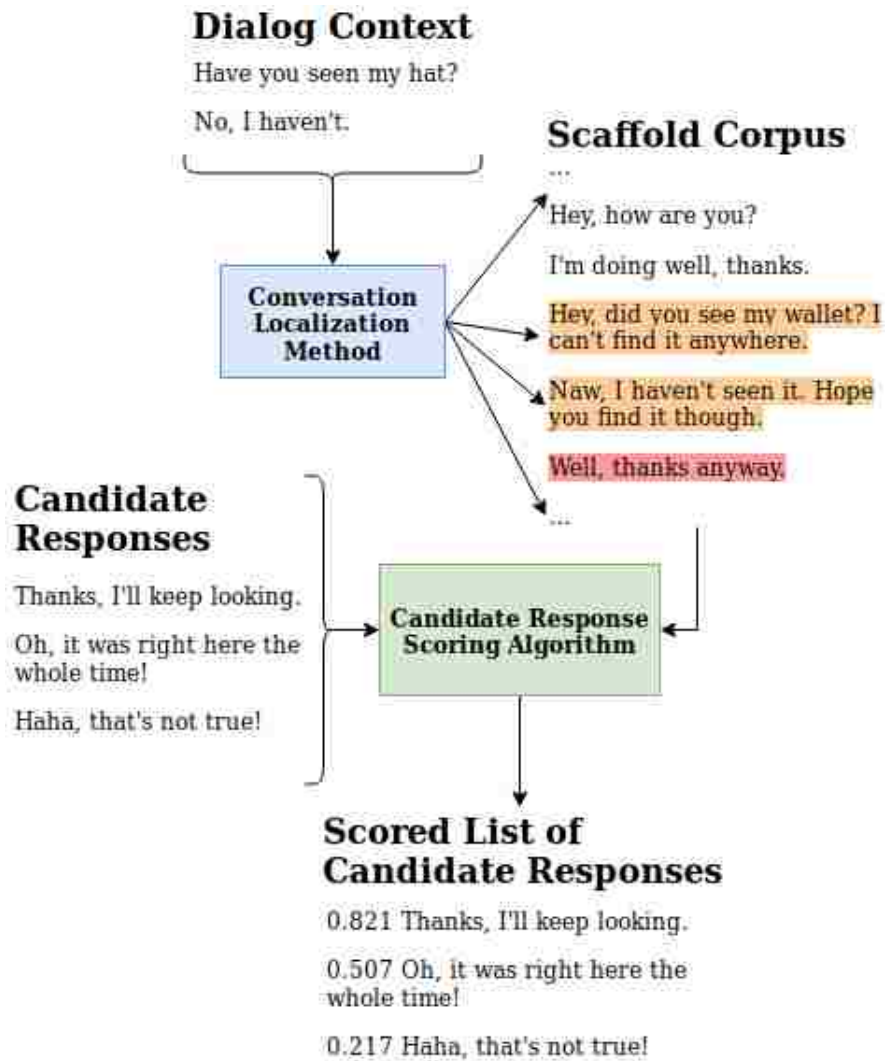


Figure 6.1: Conversational Scaffolding workflow: The dialog context is converted to an array of sentence-level linguistic embeddings using Google’s Universal Sentence Encoder, then passed to one of several localization methods in order to determine which utterances in the scaffold corpus represent the best context match. The matching contexts (orange), along with the embedded representation of their direct successors (red) within the scaffold corpus, are then passed to the Response Scoring Algorithm, which assigns a numerical value to each candidate response.

for a dialog history of length  $n$ , the optimal contextual match can be identified using the expression

$$\min_z \sum_{i=1}^n \|v_i - s_{z+i}\| \quad (6.1)$$

where  $\{v_1, \dots, v_n\}$  are the vector embeddings of the  $n$  most recent utterances in the current dialog and  $\{s_{z+1}, \dots, s_{z+n}\}$  represent the vectors located within a sliding window of length  $n$  beginning at element  $z$  of the pre-embedded scaffold corpus. The notation  $\|x\|$  represents the Euclidean norm of vector  $x$ .

This *Euclidean distance* approach is easy to calculate, but it ignores the powerful analogical structure inherent within the embedding space. For example, assume we have the following dialog history coupled with two potential contextual matches in the scaffold corpus:

**dialog history**

- |   |
|---|
| <ol style="list-style-type: none"><li>1. Did you watch the basketball game?</li><li>2. Yeah, that slam dunk at the end was really impressive.</li></ol> |
|---|

**contextual match A**

- |   |
|---|
| <ol style="list-style-type: none"><li>1. Did you watch the football game?</li><li>2. Yeah, that touchdown at the end was really impressive.</li></ol> |
|---|

**contextual match B**

- |   |
|---|
| <ol style="list-style-type: none"><li>1. Did you watch the basketball game?</li><li>2. Yeah, that was an impressive game.</li></ol> |
|---|

A Euclidean distance approach using Google’s Universal Sentence Encoder Lite and Eq. 6.1 above will select contextual match B (with a summed distance of 0.884) over A (which has a summed distance of 1.191). And yet in many ways, contextual match A is a closer semantic parallel to the actual dialog history. In particular, the conversational *pattern* exhibited in A is an almost perfect match for the dialog, even though the topic of conversation differs.

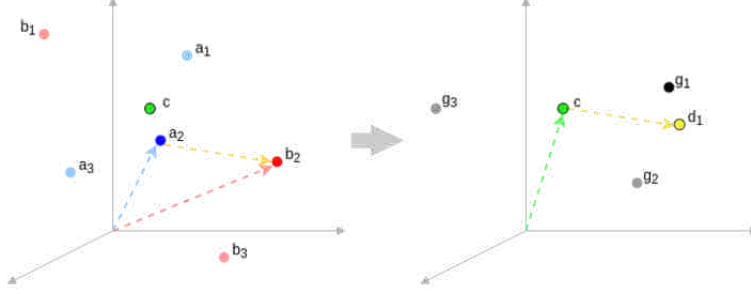


Figure 6.2: Naive Analogy, where:  $c$  (green) represents the embedded input utterance,  $a_i$  (blue) represent the nearest embedded utterances from the scaffold corpus,  $b_i$  (red) represent the embedded successors to  $a_i$  in the scaffold corpus,  $d_1 = c + b_2 - a_2$  (yellow) represents the ‘ideal’ response, and  $g_i$  (grey and black) represent embedded candidate responses with  $g_1$  (black) representing the response selected by the naive-analogy scoring algorithm.

In order to capture these subtleties, we propose two alternate methods of contextual alignment, represented in Figure 6.1 as Conversation Localization Methods:

### 1. *Difference Vectors*

This localization method seeks to capture the flow of the conversation by tracking the change in vector location between each turn in the dialogue history. The difference between each successive pair of vectors in the dialogue history is concatenated, and this concatenated vector is then aligned with a similarly concatenated vector of the differences between successive turns in the scaffold corpus. An optimal contextual match is calculated as:

$$\min_z \left\| \text{concat}_{i=1}^{n-1}(v_{i+1} - v_i) - \text{concat}_{i=1}^{n-1}(s_{z+i+1} - s_{z+i}) \right\| \quad (6.2)$$

where  $\{v_1, \dots, v_n\}$  are the vector embeddings of the  $n$  most recent utterances in the current dialog,  $\{s_{z+1}, \dots, s_{z+n}\}$  represent the vectors located within a sliding window of length  $n$  beginning at element  $z$  of the pre-embedded scaffold corpus,  $\|x\|$  represents the Euclidean norm of vector  $x$ , and  $\text{concat}_{i=1}^n(x_i)$  denotes the vector concatenation of the set  $\{x_1, \dots, x_n\}$ .

### 2. *Embedded Concatenation*

This localization method leverages the structure of the embedding space directly by concate-

Table 6.1: Dialog localization scores obtained using the dialog history and possible contextual matches shown in Section 3.1. Lower scores (bolded) indicate a better contextual match.

	naive Eucl.	diff. vectors	embedded concat.
match A	1.191	<b>0.846</b>	1.353
match B	<b>0.884</b>	0.884	<b>1.350</b>

nating the input sentences prior to encoding them via Universal Sentence Encoder Lite [17]. A naive Euclidean distance metric is then used to match the embedded concatenation against each element in the pre-embedded scaffold corpus. The optimal contextual match is:

$$\min_z ||\text{embed}(h_1 + \dots + h_n) - s_z|| \quad (6.3)$$

where  $\{h_1, \dots, h_z\}$  are the plain text (i.e. *unembedded*) utterances in the dialog history, the  $+$  symbol represents string concatenation (with an extra space inserted between sentences),  $s_z$  is an arbitrary vector located within the pre-embedded scaffold corpus, and  $\text{embed}(x)$  denotes the process of embedding a plain text utterance  $x$  to obtain its vector representation.

Table 6.1 shows the relative scores obtained when applying each of these localization methods to the dialog history and contextual match options presented above. In the case of embedded concatenation, the concatenated dialog history was compared to both sentences in contextual matches A and B, with the minimum distance being recorded in the table. (Interestingly, although the embedded concatenation method prefers match B in this example, it has an even higher preference for complex utterances such as ‘Did you watch the game last night? That slam dunk at the end was amazing’, which produces a score of 1.347.)

Note that the described localization methods assume that only a single, optimal, contextual match is desired. This was done for simplicity. In reality, it is often beneficial to take the  $k$  best matches – and in fact many of the scoring algorithms in section 6.3.2 require  $k > 1$ . In the next section, we assume a value of  $k = 3$  for clarity. In our empirical experiments, a value of  $k = 5$  was used.



### 6.3.2 Candidate Response Scoring

Once the top  $k$  contextual matches for the dialog history have been identified, the candidate responses can be scored. Candidate responses may come from a repository of pre-selected utterances, or they may be produced dynamically via generative models, scripted templates, or other text generation methods. For ease of representation, the algorithm descriptions in Figures 7.3-7.5 assume a conversation history of length 1 combined with a Euclidean distance approach to conversational localization. Extensions to longer conversation histories and more complex localization methods are straightforward and easy to implement.

In Figs 7.3-7.5, the use of the letters  $a_i$ ,  $b_i$ ,  $c$ , and  $d_i$  corresponds to the classic linguistic analogy structure  $a:b::c:d$ , which can be solved using vector offsets of the form  $c + b - a = d$ .

#### 1. *Naive Analogy* (Fig 7.3)

This scoring algorithm represents the simplest possible use of analogical structure when scoring candidate responses. Using a value of  $k = 1$ , the naive analogy locates the single best context match within the scaffold corpus, along with its (pre-embedded) successor. The vector difference between the successor and the last sentence in the context window is then added to the embedded vector representation of the most recent utterance in the dialog history. Candidate utterances are scored based on their distance from the resulting point in vector space.

Intuitively, this process can be described as seeking a candidate response that is related to the dialog history in the same way that the scaffold corpus successor is related to the sentences that precede it.

#### 2. *Scattershot* (Fig 7.4)

The scattershot scoring algorithm takes the non-deterministic nature of language into account by assuming that there are many valid responses. It therefore searches for a candidate response that matches *any* of several high-scoring context matches. In this method, the vector differences between each context match and its respective successor are calculated separately,

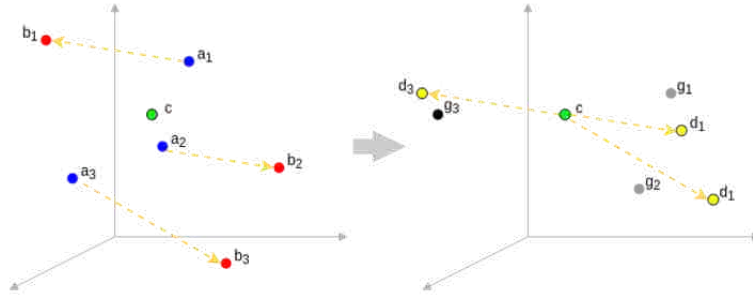


Figure 6.3: Scattershot Method, where:  $c$  (green) represents the embedded input utterance,  $a_i$  (blue) represent the nearest embedded utterances from the scaffold corpus,  $b_i$  (red) represent the associated embedded successors to  $a_i$  in the scaffold,  $d_i = c + (b_i - a_i)$  (yellow) represent the ‘ideal’ responses, and  $g_i$  (grey and black) represent embedded candidate responses with  $g_3$  (black) representing the response selected by the scattershot scoring algorithm.

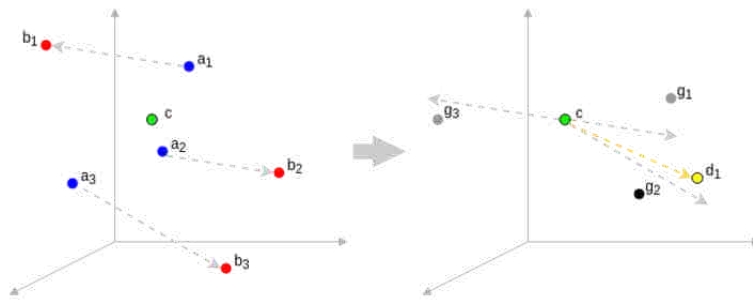


Figure 6.4: Flow Vectors Method, where:  $c$  (green) represents the embedded input utterance,  $a_i$  (blue) represent the nearest embedded utterances from the scaffold corpus,  $b_i$  (red) represent the associated embedded successors to  $a_i$  in the scaffold corpus,  $d_1 = c + 1/n \sum (b_i - a_i)$  (yellow) represents the ‘ideal’ response, and  $g_i$  (grey and black) represent embedded candidate responses with  $g_2$  (black) representing the response selected by the flow vectors scoring algorithm.

and then added to the vector embedding of the most recent utterance in the dialog history. The result is a set of  $k$  target points, each of which represents a possible valid conversational response. The candidate located nearest to *any one* of these points receives the highest score.

### 3. Flow Vectors (Fig 7.5)

Lastly, the flow vectors algorithm presumes that there is some manifold of acceptable responses within the embedding space, and seeks to calculate the centroid of that manifold by averaging the differences between multiple context matches and their successors. The candidate response nearest to this averaged centroid receives the highest score.

## 6.4 Experimental Setup

In current state-of-the-art conversational systems, candidate utterances may be obtained via generative models [143, 156], template-based algorithms [49], or pre-built language repositories [2, 171]. Our experimental setup mimics a system that uses a small ensemble of response generators to produce and/or retrieve candidate utterances. The task of the scaffolding algorithm is to leverage the dialog patterns in the scaffold corpus in order to identify the best possible response.

We also present two sample conversations generated using a much larger (ca. 20,000 utterances) set of candidates, as described in Section 6.5.2.

### 6.4.1 Text Corpora

To simulate an open-domain dialog setting, we merged data from four different sources:

1. Chit-Chat<sup>3</sup> [citation omitted for anonymity]
2. Daily Dialog<sup>4</sup> [84]
3. A 33 million word subset of Reddit<sup>5</sup>
4. The Ubuntu Dialogue Corpus<sup>6</sup> [92]

This provided us with a broad spread of conversation topics and dialog styles. The Chit-Chat dataset, developed in house by our research lab, contains 483,112 dialog turns between university students using an informal online chat framework. The Daily Dialog dataset simulates common, real-life interactions such as shopping or ordering food at a restaurant. Reddit<sup>7</sup> covers an array of general topics, with copious instances of web links, internet acronyms, and trollish behavior. Finally, the Ubuntu Dialogue Corpus contains 966,400 dialog turns taken from the Ubuntu Chat Logs, with a heavy emphasis on troubleshooting and

---

<sup>3</sup>url omitted for anonymity

<sup>4</sup><https://aclanthology.coli.uni-saarland.de/papers/I17-1099/i17-1099>

<sup>5</sup><http://files.pushshift.io/reddit/>

<sup>6</sup><https://www.kaggle.com/rtatman/ubuntu-dialogue-corpus>

<sup>7</sup>Due to the massive size of Reddit, we only used a subset of the comments and posts from June 2014 to November 2014.

technical support. The datasets vary widely in terms of vocabulary size, average turn length, and word count, as shown in Table 6.2.

### 6.4.2 Evaluation Task

To evaluate our scaffolding algorithms, we began by splitting the combined dataset into two blocks: (1) a scaffold corpus, and (2) an evaluation corpus. The agent’s task was to predict the correct follow-on sentence for each dialog in the evaluation corpus.

Table 6.2: Dataset Statistics

<b>Chit-Chat</b>	
Number of Words	8,433,086
Number of Turns	483,112
Average Length of Turns	88.86
Vocabulary Size	85,952
<b>Daily Dialog</b>	
Number of Words	3,449,782
Number of Turns	243,520
Average Length of Turns	62.85
Vocabulary Size	26,116
<b>Reddit</b>	
Number of Words	33,847,503
Number of Turns	966,400
Average Length of Turns	194.73
Vocabulary Size	434,539
<b>Ubuntu</b>	
Number of Words	15,696,635
Number of Turns	966,400
Average Length of Turns	88.33
Vocabulary Size	145,594

To set up this task, we first needed to standardize the formats of the datasets. Chit-Chat, Daily Dialog, and the Ubuntu Dialogue Corpus are all two-partner conversations, while the Reddit data was structured more like a tree with a post at the top followed by comments responding to that post and then more comments responding to other comments. In standardizing the data, we chose to ignore distinctions between actual users and flatten

the tree so that any Reddit thread was treated as a two-partner conversation between some speaker A and another speaker B.

Because some datasets, like Chit-Chat and the Ubuntu Dialogue Corpus, had many turns per conversation, we windowed our original data to create a sequence of shorter conversations with smaller dialog contexts. We chose a window size of four and a stride of one. Hence, if our original conversation had six turns, the windowed data would have three new conversations all with four turns each. We then set aside 3,311 conversations (about 5% of the smallest corpus) from each dataset to create the evaluation corpus, with the rest used as scaffolding.

Finally, the evaluation corpus was used to create a sequence of 13,244 windowed conversations. Each dialog from this evaluation set was paired with six candidate responses: (a) the correct follow-on sentence for the given dialog history, and (b) five distractors randomly chosen from elsewhere in the evaluation corpus. The scaffolding algorithms in Section 6.3.2, along with several baselines described in Section 6.4.3, were tasked with identifying the true response.

### 6.4.3 Baselines

We selected four baselines to compare against our candidate response scoring algorithms.

#### *Naive Nearest*

This algorithm is a non-analogical companion to the Naive Analogy algorithm depicted in Figure 7.3. Rather than calculating the ideal response as  $d_1 = c + b_i - a_i$ , the naive-nearest algorithm calculates  $d_1 = b_i$ .

#### *Approximate Nearest Neighbor (ANN)*

This algorithm implements an Approximate Nearest Neighbor scoring strategy. Its ideal target point is calculated in the same way as the flow vectors algorithm, but with  $d_1 = 1/n \sum b_i$ . The analogical structure of the embedding space is ignored.

### Neural Network

We also implemented a multi-layer regression network using Tensorflow [1]. As input it accepts two utterances from the dialog history, each embedded as a 512 dimensional vector using the Universal Sentence Encoder Lite. It then predicts the ideal target point as a 512 dimensional output vector. We used two hidden layers with 2048 and 2014 units respectively, with exponential linear unit (ELU) activation functions and mean squared error (MSE) loss. We found that training the network using a simple stochastic gradient descent optimizer with a learning rate of .001 worked best. We employed a 25% dropout.

### Random

This baseline randomly selects one of the candidate responses without reference to the dialog history.

## 6.5 Results

Table 6.3: Algorithm accuracy on a response prioritization task with 13,244 distinct conversations. The value  $n$  denotes the length of the dialog history used. The neural network was unable to select a response in some contexts because it requires two distinct vectors as input. The highest-scoring algorithm in each column is shown in bold-face text.

	$n=1$ Euclidean dist.	$n=2$ Euclidean dist.	$n=2$ diff. vectors	$n=2$ embed. concat.
flow vectors	.6729	.6041	.6153	.6247
scattershot	<b>.7088</b>	.5930	<b>.6399</b>	<b>.6807</b>
naive-analogy	.6751	.5616	.6188	.6229
naive-nearest	.6500	.5615	.3645	.5897
ANN classifier	.6906	<b>.6554</b>	.3771	.6496
network	n/a	.5041	.5041	n/a
random	.1699	.1713	.1660	.1606

Experimental results are shown in Table 6.3. The scattershot algorithm shows a clear advantage over other variants in most scenarios. We hypothesize that this is because it takes the nondeterministic nature of language into account, allowing the scaffolding algorithm to select a candidate that most closely matches one of many possible valid responses. For

example, the question “Have you seen any good movies lately?” can be appropriately answered by statements including “Yes”, “No”, “I saw Iron Man last night”, and “Are you crazy? You know I’m afraid of theaters.” Whereas the flow vectors or ANN algorithms would attempt to average these wildly disparate utterances, the scattershot algorithm will seek a response that lies close to one of the scaffold examples.

Additionally, we observe that the naive-analogy algorithm outperforms the naive-nearest algorithm in 3/4 scenarios, supporting the theory that response accuracy can be improved by leveraging the inherent analogical structure of the embedding space.

Surprisingly, the same pattern was not observed when comparing the flow vectors and ANN classifier algorithms. Like naive-analogy and naive-nearest, these two algorithms differ primarily in their use of analogical structure. The fact that ANN tends to outperform flow vectors suggests that in this case, the averaging of multiple offset vectors (e.g. flow vectors) results in a target point that lies far from the manifold of valid responses, whereas the averaging of multiple actual sentence embeddings (e.g. ANN) remains closer to the ideal manifold. Further research is needed to understand this phenomenon and quantify the complex structures found in semantic embedding spaces.

Lastly, we observe that a smaller dialog history ( $n = 1$ ) seems to be most effective in terms of facilitating accurate responses. This seems counter-intuitive, since many dialog scenarios include utterances like ‘yes’ or ‘of course not’, which are difficult to respond to appropriately unless a larger context window is used. This drawback appears to be overshadowed by the relative ease of matching a context window of size 1 to the scaffold corpus, as opposed to attempting to find a set of sentences that match. In much the same way that Mikolov et al. [96] argued that a computationally simple system, by leveraging more data, can produce a superior model, we hypothesize that a smaller context window, by enabling closer context matches, facilitates improved response accuracy.

Table 6.4: Confusion matrix showing how well each dataset, when used as a scaffold corpus, is able to select appropriate responses for dialogues drawn from the other corpora. Each column contains a scaffold corpus, each row an evaluation corpus. The scattershot algorithm was used in conjunction with the embedded concatenation localization method, with a dialog history of size  $n = 2$  and with 3,311 evaluation dialogues drawn from each corpus. The highest accuracy level in each column is shown in bold-face text.

	Chit-Chat	Daily Dialog	Reddit	Ubuntu
Chit-Chat	.5826	<b>.6563</b>	.6068	.3455
Daily Dialog	.5421	.6540	.5847	.3582
Reddit	<b>.5886</b>	.6261	<b>.7508</b>	.3860
Ubuntu	.4971	.4666	.5639	<b>.7523</b>

### 6.5.1 Generalization Across Datasets

We found ourselves curious as to what extent each dialog corpus was able to generalize to the other corpora in the evaluation set. We therefore implemented the following experiment: Using the scattershot algorithm and embedded concatenation localization method, we created a confusion matrix showing how well the algorithm performed when using only one of our four corpora as its scaffold.

Results are shown in Table 6.4. The unusually high performance seen when both the scaffold and evaluation corpus were taken from the Ubuntu dataset can be explained by the high level of overlap within the Ubuntu dialog corpus. An investigation of the data downloaded from Kaggle reveals that between the three files (“dialogueText\_301.csv”, “dialogueText\_196.csv”, and “dialogueText.csv”) there was an overlap of 53.35% in the original data (14,318,055 non-unique turns out of a total of 26,839,031 turns). As a result, the evaluation corpus drawn from the Ubuntu dataset contained exact copies of dialogues in the Ubuntu scaffold corpus. The other corpora had little or no overlap.

### 6.5.2 Retrieval from Larger Data Repositories

To examine algorithm performance on large-scale retrieval tasks, we set up a system in which a human typed an opening sentence and we then used different variants of our algorithms as a retrieval method on the scaffold corpus. We permitted one augmentation to our algorithms



for this experiment: candidate responses that were too similar<sup>8</sup> to the most recent statement in the dialog history were excluded from consideration. This constitutes an extension at the sentence level of the traditional exclusion of source words when solving analogical queries via word embeddings [56, 97]. Without it, the scaffolding algorithm tends to select sentences that parrot or reflect the content of the dialog history.

Two typical conversations generated using this method are shown below. Human-generated text is preceded by the > symbol.

(scattershot,  $n=2$ , embedded concatenation,  $\tau = 0.6$ )

> Hello.

Hey!

> How are you doing today?

How are you?

> I am good. How has your day been so far?

I've been pretty good, I was sick earlier, but better now.

> Oh man, that stinks. What were you sick with? I am glad you are feeling better though!

I had like ten servings of it.

> Uh oh. That sounds like a lot. Was it good food?

I had such tasty food too.

> Yum. What is your favorite food?

yum sounds so good right now haha! I love every fruit! I like rice and pickles too haha. but like you I like just about everything

> I love fruit too. So, have you seen any good movies lately?

Black panther is so good!!!

> Yeah I really liked it too. I think that or, Thor Ragnarok might be my favorite Marvel movie.

No , I haven't . This is my first time to watch it .

> Oh yeah?

\*It flips him off.\*

> haha ok. So do you like to read?

I do! I'd appreciate it.

> What are some of your favorite books?

---

<sup>8</sup>Similarity was defined as Euclidean distance  $< \tau$ , where  $\tau$  is a hand-selected threshold value.

It's a little bit of romance and comedy.

> Cool. Well, I have to go. It has been nice talking to you!

Okay ! Bye !

(scattershot,  $n=2$ , embedded concatenation,  $\tau = 0.6$ )

> Hey. What's up?

Hey, whats up?

> Not much. How is your day going so far?

Hi, it's good!

> Good to hear. What are you working on?

I'm figuring out my budget .

> Oh that sounds fun. What are you going to do after you finish?

Well I plan to get into the accounting program

> Oh cool! Do you like accounting?

I've thought about going into cardiology, that seems interesting, but also I'm considering anesthesiology.

> Oh wow. Those are both very different from accounting. Are they hard?

semester for you and me. Is your next semester seeming to look like its going to be a little easier?

> I think this semester will be fun but a little challenging.

That it does! Even though you may not be getting everything out of your classes now but oh well...;)

> Do you think you will have any time to go to see a movie this semester?

That it does! Even though you may not be getting everything out of your classes now but oh well...;)

> So, where are you from originally?

Haha well I'm from Arkansas

> Oh cool. I have never been there. Do you like it?

Ya actually I did enjoy the shreveport one when I lived there last year (2012 Mardi gras) but you're right, it's almost nothing Compared to most of the NO ones. Still very enjoyable. My of my friends lives right on the route so it was nice having somewhere to go for it lol

> haha fun. Do you like it here then?

Cool

> If you could vacation anywhere, where would it be?

well,its like 1 am here,i should get to bed, it was nice talking to you, have a great day.

> Oh wow. Goodnight. Talk to you later.

Thanks. You too!

## 6.6 Conclusions and Future Work

We have presented a response prioritization method using pre-trained sentence embeddings that (a) leverages the inherent analogical properties of the embedding space and (b) accounts for the frequently non-deterministic nature of natural language conversations. In experiments using dialog contexts of length 1 and 2, our scattershot algorithm exceeds the performance of a random baseline by a factor of 4 and out-performs both an Approximate Nearest Neighbor classifier and a neural model trained to predict the next sentence in a conversational sequence. We also demonstrate that our scaffolding algorithms are able to produce relevant and engaging conversations when retrieving utterances from large sentence repositories.

Future work in this area should explore the effect of using alternate embedding models such as skip-thought vectors [74], quick-thought vectors [89], Universal Sentence Encoder Large [17], InferSent [23], InferLite [71], or bag-of-words averaging [5]. The impact of dialogue context length on classification accuracy should be further investigated, along with the possibility of hybrid algorithms that use short dialogue histories unless suitable heuristics indicate that a longer context is needed. Finally, more work should be done on understanding the analogical structure of existing sentence-level embedding spaces, so that this structure can be more effectively leveraged in real-life applications.

## Chapter 7

### **BYU-EVE: Mixed Initiative Dialog via Structured Knowledge Graph Traversal and Conversational Scaffolding**

by Nancy Fulda, Tyler Etchart, William Myers, Daniel Ricks, Zachary Brown,  
Joseph Szendre, Ben Murdoch, Andrew Carr and David Wingate,  
Amazon Alexa Prize Proceedings, November 2018.

#### **Abstract**

We present BYU-EVE, an open domain dialogue architecture that combines the strengths of hand-crafted rules, deep learning, and structured knowledge graph traversal in order to create satisfying user experiences. Rather than viewing dialogue as a strict mapping between input and output texts, EVE treats conversations as a collaborative process in which two jointly coordinating agents chart a trajectory through experiential space. A key element of this architecture is the use of conversational scaffolding, a technique which uses a (small) conversational dataset to define a generalized response strategy. We also take the innovative approach of integrating the agent’s self and user models directly within the knowledge graph. This allows EVE to discern topics of shared interest while simultaneously identifying areas of ambiguity or cognitive dissonance.

## 7.1 Overview

### 7.1.1 Open Domain Dialog: Challenges and Opportunities

As demand for voice technology expands, the challenges inherent in conversational AI become more pressing. Users desire voice assistants who behave less like machines and more like humans [14, 121, 135]. They don't simply want to query their devices; they seek to engage with them in complex exchanges, use them as mental sounding boards, and receive social validation in response to their statements. This type of interaction encompasses, but also goes beyond, database queries and question/answer models. In order to engage with the user fully, the system must understand the rhythms of conversational flow, maintain an internal identity and provide information that is geared to satisfy the unspoken desires of the user.

Our architecture is designed to accomplish all three of these tasks. We address conversational flow via the scaffolding technique presented in Section 7.5.1. Internal identity is maintained via specially tagged knowledge graph nodes as described in Section 7.3, and targeted information retrieval is discussed in section 7.7.3. Additionally, we have structured our model to sustain expansion into unrestricted dialog settings: rather than being an expert in a few specialized areas, our infrastructure is designed to apply general-purpose knowledge across a wide variety of topics and conversational settings.

### 7.1.2 Invisible Walls vs. Infinite Horizon: How Open Is 'Open'?

Arbitrary social dialog is a challenging problem, yet highly promising, with potential applications in areas such as education [85], political activism [57, 79], and elder care [37], among others. In order to manage the complexities inherent in unstructured, free-form dialog, many conversational systems employ an 'invisible wall' technique. The system gives the appearance of being able to discuss any topic, but is in reality guiding the conversation only to those topics that it is able to handle well. This technique has been successfully employed by [120, 161].

To make significant progress in this domain we seek to go further. Following the example set by [59], we envision an ‘infinite horizon’ scenario in which the system is able to navigate the full range of human conversation. This does not imply that the system must be omniscient. Rather, it means that the system is able to apply the knowledge at its disposal within a broad range of conversational contexts.

This is accomplished by housing the system’s core knowledge in a single location. Instead of querying myriad APIs via dozens of specialized response generators, our architecture accumulates information within a centralized knowledge graph. Individual response generators navigate this graph in structured ways, creating a flexible, easily scalable system. In this paradigm, expanding the socialbot to handle new topic areas becomes a simple matter of augmenting the knowledge graph; no hand-coding and no new response generators are required.

We hasten to note that vision and reality are often two separate things. EVE was produced as part of Amazon’s Alexa Prize Challenge, with corresponding deadlines that required her to enter production before the core structures were complete. As a result, the system is currently a hybrid between our core architectural ideas and specialized, topic-specific response generators that use direct API calls rather than relying on the central knowledge graph. These were created to satisfy consumer demand until the centralized infrastructure is finished.

### **7.1.3 Text Retrieval vs. Text Generation: Between a Rock and a Hard Place**

Ultimately, a dialog system exists to generate text. This text may be retrieved from a curated and/or web-scraped repository [77, 120], generated via neural network models [77, 163], or constructed using a combination of templates and placeholders [77].

Each of these methods has drawbacks. In open domain dialog, the scope of possible conversation topics makes retrieval-based systems unfeasible; there are simply too many conversational pathways. On the other hand, templates tend to sound repetitive and robotic,

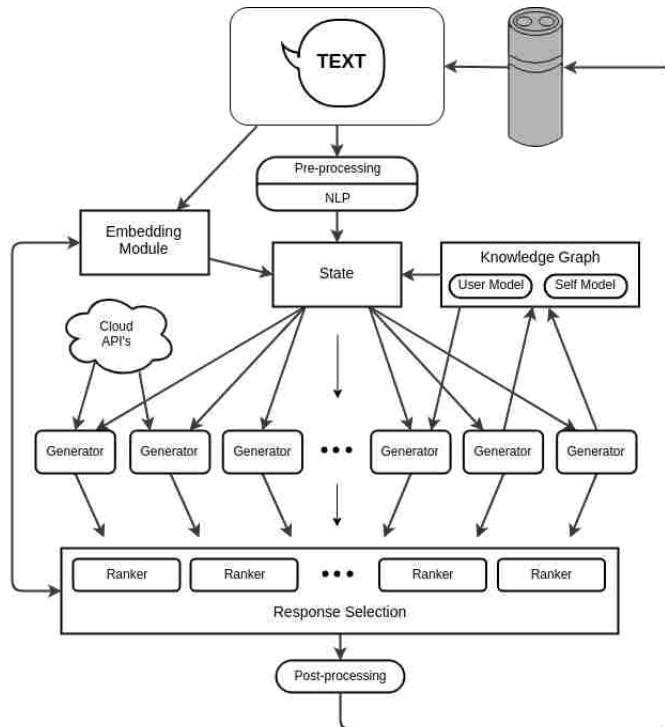


Figure 7.1: Our architecture is built on the premise of multiple candidate response generators vying for the attention of the dialog manager. Each generator produces a specific type of response: emotive utterance, fact retrieval, conversation offer, self-analysis, etc. The dialog manager analyzes these outputs, then selects and combines the best of them to produce a fluid, natural-sounding response.

while generative models suffer from a lack of coherence, lack of consistency, and frequent lack of propriety.

So what’s a bot to do? As an interim solution, EVE uses a combination of templates and retrieval methods, but the long-term goal is to create generative models conditioned on conversational context and knowledge graph data. In other words, we wish to control *what* the system says while using a neural network to decide *how* the idea should be expressed.

#### 7.1.4 Choosing What to Say: A Question of Composition

When preparing final output for the user, we rely on the insight that conversation is a multi-channel communication stream along which not only information, but also emotive and psychological data is being passed. For example, when Alice tells Bob that she failed her math test, Bob’s response is likely composite: “Oh no!” he might say, “Will you get a chance to re-take it?” The first statement acknowledges and empathizes with Alice’s emotional state, while the second propels the conversation forward with a request for information.

We model this compositionality by classifying candidate responses into communication ‘channels’. For each user utterance, our system attempts to generate (a) an emotive response, (b) an informative reply to the user utterance, and (c) a conversational offer such as a follow-up question or proposed conversation topic. To simplify terminology, we refer to these as emotes, answers, and offers. An ensemble of ranking algorithms scores each candidate response, and the highest-ranking emote, answer, and offer respectively are combined into a final output response.

A key element of this process is the idea of *conversational scaffolding*, the principle that a relatively small set of example conversations can be leveraged to determine which candidate responses are viable. This is accomplished by leveraging the analogical properties of embedding spaces at the conversational level, as described in Section 7.5.1.



## 7.2 Knowledge Representation

The storage, retrieval, and representation of knowledge is a core issue when it comes to tackling open-ended conversation. Without some form of persistent memory, it is not possible for a conversational AI to progress beyond simple social pleasantries or, at most, a generative or probabilistic model that reproduces common factual statements. On the other hand, relying on highly specialized memory retrieval mechanisms, such as an IMDb API for movies, an ESPN API for sports, or other topic-specific cloud APIs, makes it difficult to synchronize knowledge across response generators or to find relationships that span multiple topics. A unified, persistent memory allows for intelligent topic switching and strong contextualization of conversation.

While many developers are more familiar with relational database management systems (RDBMS), such as MySQL, they are not the right pick for representing general knowledge, mainly due to their rigidity. Trying to come up with tables that make sense for actors, sports, towns, books, planets, etc. is intractable; the tables would end up tremendously sparse. Graph databases, on the other hand, lend themselves to these sorts of free-form ideas. In addition, relationships between two ideas is a first-class citizen with graph databases, whereas relationships in relational databases typically require an expensive join. When representing knowledge, understanding and traversing the relationship between two ideas is crucial for open-ended conversation.

In light of these factors, we have chosen to implement persistent memory in the form of a graph database (a.k.a. knowledge graph) consisting of unique nodes connected by labeled edges. Our knowledge base currently contains 36,367,345 nodes and 164,664,878 edges reflecting a wide array of topics such as literature, biology, genetics, visual media, manufacturing, biographical information, and many others.

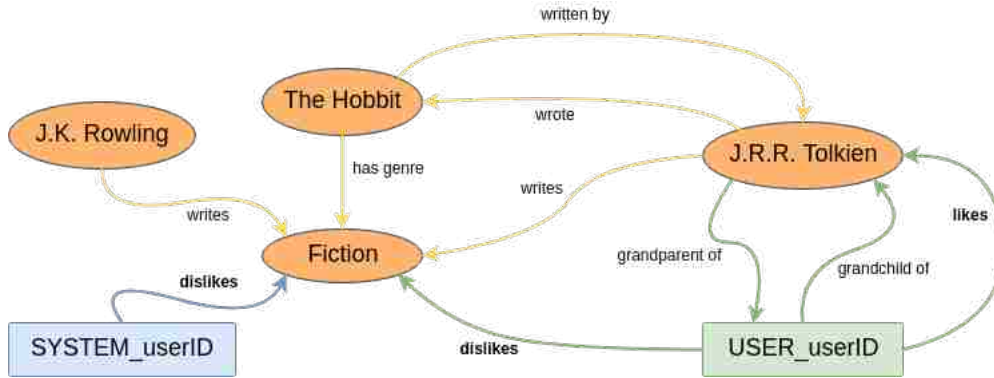


Figure 7.2: A representation of a portion of our knowledge graph. Orange nodes and edges correspond to general facts contained in the knowledge graph, the green node and edges represent a model of the user associated with a specific userID, and the blue node and edge represent a portion of the system model corresponding to the same userID. Bolded edge labels represent facts of interest that elicit responses from the MCI and MCE algorithms discussed in Section 7.3.1.

### 7.2.1 Property Graph vs. Resource Description Framework (RDF)

There are two main competing types of graph databases, property graphs and resource description frameworks (RDF).

**Property Graph** A property graph can best be described as a free-form web of information with a structure implied by its name. Nodes (or vertices) of the graph consist of labeled items such as “J.K. Rowling”, “science fiction”, or “dalmation”. These nodes can have properties attached: Thus, one property for “J.K. Rowling” might be an alias property with the value “Joanne Kathleen Rowling”. Some properties are important enough to deserve their own node. In these cases, the property itself is considered an edge (and will be referred to as such from here on out) and the edge’s value is a pointer to another item in the graph. For example, the edge “author of” would point from the “J.K. Rowling” node to the node labeled “Harry Potter and the Sorcerer’s Stone”. In turn, *that* node might have an alias property with the value “Harry Potter and the Philosopher’s Stone”, as well as a genre edge pointing to the “fantasy” node. Storing knowledge in this way creates a flexible structure that keeps related ideas close to one another via edges.

The main benefit of using the property graph variant is the loose structure and the ability to query it using the Apache Tinkerpop Gremlin language [41]. Gremlin allows the user to make complex path queries that allow multiple degree jumps in a single query. We found this extremely useful when writing generators that wanted to tap into the knowledge graph as it allowed them to make multiple connections to a single idea, much as a human might.

**RDF** RDF databases are at heart, triplet stores. They come in subject-predicate-object format, such as “J.K. Rowling”-“author of”-“Harry Potter and the Sorcerer’s Stone”. Storing the data in triplet form make queries very quick, but limited. SPARQL [118] is the main query language for RDF databases and as eloquently put by Oracle: “The SPARQL query language is intended primarily for pattern (subgraph) matching rather than path traversal. ... SPARQL 1.1, however, still lacks the ability to reference a path directly in a query (e.g., the ability to store the result of a property path query in a path variable). Without this ability, it is not possible to match an arbitrary length path and return the path itself or perform operations based on characteristics of the path, such as path length.” [27].

Even though most open knowledge graphs, like Wikidata and Conceptnet, come in RDF format and RDF queries are on average faster than property graph queries [3], we found the path traversal feature too important to pass up. As a result, we built a pipeline that converted the RDF Wikidata graph into a property graph so that we could take full advantage of the path traversal features offered by Gremlin.

### 7.2.2 Wikidata

Our core knowledge graph content is based on Wikidata [159]. Wikidata is a Wikimedia project that seeks to structure the world’s knowledge. It can be thought of as Wikipedia in knowledge graph format. We originally based our knowledge graph on Conceptnet [? ], but found that it was not large enough to facilitate open-ended conversation. We then turned

to Wikidata, which we first had to repurpose by stripping out all non-English ideas. We then stripped out nonconversational nodes like “Wikimedia Disambiguation”, and finally we added Google’s Universal Sentence Embeddings [17] for the descriptions of each node. Even with all the cleaning, we found that Wikidata was somewhat esoteric in the selection facts surfaced, resulting in statements like “Do you consider cat to be a typical example of the animal painting genre?”. While this problem persists today, we were able to greatly minimize it by selectively traversing only edges that make sense in the context of the current topic, i.e., when talking about movies we want to traverse “acted in” edges and not “subclass” edges.

### 7.2.3 Knowledge Graph Hosting

Our knowledge graph is managed via the Amazon Neptune service, a choice which simplifies load balancing, fault tolerance, and general updates to the graph. We use the Apache Tinkerpop Gremlin language [41] (the python variant [40]) for queries.

## 7.3 User and Self Modeling

In order to approach human-level conversation, the system must not only remember what has been said in exchanges with users, but it must *interpret* what has been said. This means the system must understand how past utterances relate to each other, to each of the conversation participants, and to potential future utterances.

We address this problem by modeling both the user and the system as unique nodes within the knowledge graph (see Figure 7.2). Edges attached to these nodes represent the core ideas expressed in previous utterances, such as affinities, disinclinations, and factual data including names and nicknames. As the conversation plays out, the system extracts facts from what is said by both itself and by the user and stores those facts in the knowledge graph, connecting them to appropriate nodes.

For example, if the user says “I like comics”, the system connects the user’s node with a ‘likes’ edge to a ‘comics’ node. Conversely, if the system generates an utterance expressing

an affinity, such as “I enjoy Superman comics”, the system connects its own knowledge graph node to a “Superman” node with a ‘likes’ edge.<sup>1</sup>

In this way, the system generates its own personality dynamically as a result of its interactions with the user. Because each node representing the system is user-specific, the problem of finding a personality for the system that makes a strong majority of users happy is largely mitigated, as each individual user molds their personal system model by what they choose to discuss with it.

We find this approach both novel and valuable because it places user utterances in context with past statements as well as with general world knowledge encoded in the knowledge graph. It also allows the possibility of self-reflection and conflict resolution on the part of the system. For example, if the system were to detect that the user likes “J.K. Rowling” but dislikes “Harry Potter and the Chamber of Secrets”, this might prompt it to inquire more deeply as to why the user dislikes that particular book when they are generally well-disposed toward its author.

Similarly, the system might seek to offer new conversational topics based on areas of common interest. For example, if the user likes “Star Trek” and the system likes “Space Odyssey 2001”, both of which are characterized as “science fiction”, then the system might offer new conversation topics that fit into the same category. Alternately, if the user has expressed an affinity for “burritos” and “pizza”, but a dislike for “spinach”, then the system might seek to explore conversational options that are closely related to the first two nodes but relatively distant from the disliked node; see Section 7.7.1 for more details.

### 7.3.1 Current Implementation of Dynamic Knowledge Graph Models

The current system architecture relies on two response generators to handle user- and self-modeling functionality: DataBot and ModelQueryBot. The purpose for dividing the task of

---

<sup>1</sup>One might wonder how or why the system’s response generators would express an affinity that is not already present in the knowledge graph. At present, this occurs as a result of stochastic algorithms that seek to reflect the interests expressed by the user, as well as from topic-specific (i.e. non-knowledge graph) generators that express affinities while introducing new topics for discussion.

user- and self-modeling among two separate generators is primarily due to time constraints on how quickly the generators must return a response to the user and the latency (usually about 50-100ms database + 50-300ms network) involved in querying and updating the Neptune Knowledge Graph API.

**Populating Models - DataBot** DataBot is designed to extract as many knowledge graph facts about the user and the system as possible from the conversation history and to use those facts to populate the user and self models. As of the writing of this paper, DataBot uses a combination of named entity recognition and regular expressions to search for three specific edge types: ‘likes’, ‘dislikes’, and ‘called’. ‘Called’ edges represent the name of the user or the system. ‘Likes’ and ‘dislikes’ edges represent the stated preferences of the user and the system.

Once a potential edge is found, Databot examines the utterance to identify its associated nodes and pushes a new fact of the form (node1, edge, node2) - which we will refer to as a ‘triple’ - to the appropriate model within the knowledge graph. The new triple is immediately visible to all other response generators within the system, which can use the stored information as they please.

**Employing Models - ModelQueryBot** ModelQueryBot’s design is twofold: (1) it ensures that no contradictory information exists in the system’s user and self models, and (2) it generates novel responses using those models. To accomplish the first objective, ModelQueryBot employs an algorithm which we call Model Error Detection (MED) to identify contradictory information in the user and self models based on predetermined rules. One such rule is that the user model should never have a ‘likes’ edge and a ‘dislikes’ edge connecting the same two nodes in the knowledge graph. If any contradictory information is identified, ModelQueryBot deletes the contradictory information and prompts the user for clarification in a generated response.

---

**Algorithm 4 - Model Connection Identification (MCI)**

---

**Inputs:**

$$K = (N, E, L, T)$$

$$N \subset R$$

$$E \subset R$$

$L = \text{range}(f)$  where  $f : (N \cup E) \rightarrow L \subset S$  Labels, mapping Nodes  $N$  and Edges  $E$  into natural language  $S$

$$T = \{t | t = (n_i, e_j, n_k); n_i, n_k \in N; e_j \in E\}$$

$$M_1 \subset K, M_1 = (N_1 \subset N, E_1 \subset E, L_1 \subset L, T_1 \subset T)$$

$$M_2 \subset K, M_2 = (N_2 \subset N, E_2 \subset E, L_2 \subset L, T_2 \subset T)$$

$$n'_1 \in (N \cap M_1)$$

$$n'_2 \in (N \cap M_2)$$

$X$  = A set of pre-determined epsilon pairs of interest

$R$  = A set of 'referenced' pairs of triples

**Parameters:**

$\beta \in Z$  The maximum number of triples in returned sets of triples to be resolved

**Output:**

$D$

$$A \leftarrow \{a = \{t_1, t_2\} : t_1, t_2 \in T' \text{ satisfy } n'_1 \in t_1, n'_2 \in t_2, \text{ and } (e_u, e_v) \in X \text{ for any } e_u, e_v \in E \text{ s.t. } e_u \in t_1 \text{ and } e_v \in t_2\}$$

$$A' \leftarrow A - (A \cap R) \quad b < \beta \quad D_b \leftarrow \{d_b = \{t_0, t_1, \dots, t_{b-1}, t_b\} : t_{b-1} = (n_i, e_j, n_k) \in d_{b-1} \in D_{b-1}, t_b = (n_k, e_y, n_z), i \neq z, D_0 = A'\}$$

$$D'_b \leftarrow \{d'_b = \{t_0, t_1, \dots, t_{b-1}, t_b\} : \{t_0, t_b\} \in A, d_b \in D_b\} \quad D \leftarrow (D \cup D'_b) \quad D$$

---

If no contradictory information has been found, ModelQueryBot will generate a response based off of the user and self models using algorithms we introduce as Model Connection Identification (MCI) and Model Content Extension (MCE). Using MCI and MCE, ModelQueryBot first examines the primary nodes and edges directly connected to the user and user-specific system nodes, as well as all of the secondary nodes and edges directly connected to the primary nodes. ModelQueryBot then uses one of the identified triples to populate one of several relevant, predetermined templates; for example, one such template reads as “Oh, you don’t like (node1) and since (node1 edge node2) do you also not like (node2)? I’m rather fond of (node2)”. As the user and self models grow in complexity, so does the level of reasoning apparent in the responses output by ModelQueryBot.

**Model Connection Identification (MCI)** MCI is an algorithm for finding all of the connections between two nodes in a knowledge graph within a given search radius. MCI takes the general knowledge graph  $K = (N, E, L, T)$  as input, where  $N$  is a set of indices corresponding to nodes,  $E$  is a set of indices corresponding to different edge labels,  $L$  is the set of labels that correspond to the indices in  $N$  and  $E$ , and  $T$  is the set of all fact triples of the form (node1, edge, node2) in the knowledge graph. MCI also requires two sets of triples that model entities (either distinct or identical) which we refer to as  $M_1$  and  $M_2$

as input, as well as the nodes  $n'_1$  and  $n'_2$  that represent the entities themselves. Finally, a set of pre-determined epsilon pairs of interest  $X$  (in ModelQueryBot, it is often the case that  $X = \{(likes, likes), (likes, dislikes), (dislikes, likes), (dislikes, dislikes)\}$ ), and a set  $R$  of pairs of knowledge graph triples that have previously been addressed with the user are input into MCI. Using its inputs as well as a parameter  $\beta$  to control the depth of its search, MCI collects a set  $D$  of sets of triples from the knowledge graph that connect  $n'_1$  and  $n'_2$  together which is then returned as output.  $\beta = 2$  is the maximum search window used in ModelQuerybot currently. We define MCI more formally in Algorithm 1.

**Model Content Extension (MCE)** MCE is a special use-case of MCI; we write MCE as its own algorithm here to highlight a few of the various uses for MCI that go beyond merely finding relevant factoids relating two specific nodes which can then be used to generate text. In MCE, MCI is given  $K$ ,  $X$ , and  $R$  as normal (currently with the parameter  $\beta = 2$  often kept constant as well), but then receives only one model  $M$  of a particular entity and the entity’s associated node  $n'$ . By performing MCI on only one entity, instead of trying to connect two arbitrary entities as in MCI, MCE effectively tries to find logical connections within the nodes of  $M$ ; in essence, we currently use MCE to try and find causal relationships between the “likes” and “dislikes” of the user (and potentially to give causal relationships for the “likes” and “dislikes” of our own system in future work). Therefore, ModelQueryBot uses MCE to detect particularly interesting subtleties in its model of the user - for instance, MCE would be used to ask why a user would tell us that they both “like ice cream” and “dislike dairy products” at the same time, or to ask the user if the reason they “like J.R.R Tolkein” is because they also “like the Hobbit”. MCE is expressed formally in Algorithm 2.

**Model Error Detection (MED)** MED is the algorithm we use to remove contradictory facts from our user and self models. Similarly to MCE, MED takes as input the Knowledge Graph  $K$ , a single entity model  $M$  and its associated entity node  $n'$ , as well as  $X$ , which in this case represents a set of pre-determined contradictory epsilon pairs that should



---

**Algorithm 5** - Model Content Extension (MCE)

---

**Inputs:**

$K$  The Knowledge Graph; see the description of  $K$  in Algorithm 1 - MCI for more details  
 $M \subset K$   
 $n' \in (N \cap M)$   
 $X$   
 $R$  = A set of ‘resolved’ pairs of triples

**Parameters:**

$\beta \in Z$

**Output:**

$D$

$D \leftarrow MCI(K = K, M_1 = M, M_2 = M, n'_1 = n', n'_2 = n', X = X, R = R, \beta = \beta)$

---

never be found connecting the same two nodes simultaneously (currently, MED is given  $X = \{(likes, dislikes), (dislikes, likes)\}$  in ModelQueryBot). Furthermore, MED takes in the set  $X'$  which represents a set of pre-determined ‘singular’ epsilons, which should connect a given node (in the case of MED, this node is  $n'$ ) to only one other unique node; in ModelQueryBot, MED receives  $X' = \{called\}$ . Given its inputs, MED returns a set  $C$  representing all of the ‘contradictory’ knowledge graph triples in the input Model. MED is expressed formally in Algorithm 6.

---

**Algorithm 6** - Model Error Detection (MED)

---

**Inputs:**

$K$  The Knowledge Graph; see the description of  $K$  in Algorithm 1 - MCI for more details  
 $M \subset K$   
 $n' \in (N \cap M)$   
 $X$   
 $X'$

**Output:**

$C$

$C_{pair} \leftarrow MCD(K = K, M = M, n' = n', X = X, R = \emptyset, \beta = 1)$   $C_{singular} \leftarrow \{p = \{t_1, t_2\} : t_1 \cap t_2 = \{n', e_j\} \text{ with } e_j \in X', \text{ and } t_1, t_2 \in T\}$   $C \leftarrow (C_{pair} \cup C_{singular})$   $C$

---

### 7.3.2 A Note on Privacy

When users interact with our system, they implicitly agree to share personal information (e.g., opinions, facts about themselves, etc.) in the form of natural conversation. Our system attempts to harvest this information in a way that will mold and personalize the experience for each user by adding it to the general knowledge graph. However, if at any point this becomes an issue or a user wishes to have their information deleted, it is trivial to delete

the system node and user node associated with their ID, which will in turn delete all edges associated with them and effectively remove any personal information collected.

## 7.4 Response Generation

EVE’s response generators fall into three broad categories: *Emotive generators*, which seek to identify and empathize with the user’s mood, *Knowledge Graph generators*, which seek to find connections between user utterances and related nodes within the knowledge graph, and *API generators*, which query local repositories or online data repositories in order to respond to user inquiries. We also maintain a small set of *Fallback generators* designed to engage when the system fails to find an appropriate response in one of the other categories and a set of *Scripted generators* that engage when predefined text or sensitive topic material is detected.

Generators are able to pass meta-data back and forth to one another between responses. For example, if a response generator makes use of a keyword uttered by the user, it can pass the keyword in the meta-data so other generators know what the topic of conversation was. Response generators can also request priority from the dialog manager and preserve internal state between dialog turns.

Long-term, it is expected that most API generators will be replaced with knowledge graph generators; possible exceptions being news and social media-based generators. At that point, API queries would be used primarily to populate the knowledge graph with new data.

### 7.4.1 Emotive Generators

Our emotive generators seek emotional cues in the text and attempt to respond appropriately. For example, if the user says “I like fluffy bunnies”, the SmartEmote generator might respond “No kidding? Me too!” Alternately, if the user makes a statement like “I feel sad”, the EmoBot generator will engage this emotion with a statement like “I’m sorry. You deserve to feel better than that.”

## 7.4.2 Knowledge Graph Generators

The knowledge graph generators are the core of our infrastructure, and are still in active development. DataBot and ModelqueryBot are described in Section 7.3.1. The remaining knowledge graph generators are as follows.

### **FirstDegreeFact**

Takes a node of interest and finds all relevant (by topic) first degree connections. Randomly samples a connection and then uses a template to talk about the fact.

### **FirstDegreeConversationStarter**

Similar to the FirstDegreeFact except that it takes the first degree connection and tries to transition the conversation in a relevant direction.

### **SecondDegreeFact**

Works just like the FirstDegreeFact, with the main difference of using both the first degree and second degree connections instead of just first degree connections.

The strength of the knowledge graph generators is perhaps best illustrated by EVE's handling of the 'Literature' topic. Our infrastructure does not use any book-based APIs or knowledge repositories, and yet, using only the knowledge graph generators and a small set of hand-coded conversation starters, the system is able to respond to book-related utterances so well that 'Literature' is one of our highest-rated conversation topics.

## 7.4.3 API Generators

As discussed in Section 7.1.2, the API generators were implemented primarily as placeholders to satisfy customer demand until the knowledge graph infrastructure is fully on-line. They use an elementary combination of conversation templates combined with API calls to OMDb, TMDb, ESPN, Washington Post, DuckDuckGo, Wikipedia, and Amazon Evi. Many of the API generators maintain an internal state regarding the current conversation topic, which questions were recently asked, etc.

## 7.5 Response Evaluation

Free-form conversation is variational, not deterministic; this means that there is no single correct response to a given utterance or sequence of utterances (although there are certainly plenty of incorrect ones). Rather, the set of acceptable and/or excellent responses can be modeled as a probability distribution across the range of possible utterances.

For example, the question “Have you seen any good movies lately?” can be appropriately answered by statements including “Yes”, “No”, “I saw Iron Man last night”, and “Are you crazy man? You know I’m afraid of theaters.” Each of these responses is equally valid, although not equally likely. In contrast, responses such as “copper filings”, “The United States was founded in 1776”, or “My flashlight batteries are empty” are so unlikely as to be implausible.

Within the context of conversational AI, this is both good and bad news. It’s good because our response selection criteria need not concern itself with finding the *most correct* response (because many possibilities are equally correct); It is sufficient to identify the subset of possible responses that are relatively likely to occur. From these, a final response can be selected based on criteria such as length, uniqueness, and literary quality.

Unfortunately, the variational nature of conversation makes it difficult for a naive neural network, or even a well-structured variational neural network, to learn a pattern of correct conversational responses. We address this problem by introducing *conversational scaffolding*, a technique that allows a small set of sample conversations to guide the bot’s overall behavior.

### 7.5.1 Conversational Scaffolding: An Analogical Approach to Response Prioritization

Most practitioners of natural language understanding are familiar with the analogical coherence demonstrated by word2vec [96], GLoVe [116] and other word-level embedding spaces. The basic principle is simple: (1) Take a word. (2) Convert it into a vector using a neural

network trained based on word co-occurrence patterns. (3) Now examine its relationship to other words in the embedding space. Mikolov et al. have shown and other researchers have confirmed that within these embedding spaces, simple mathematical operations are sufficient to solve analogical queries [45, 48? ?].

Our conversational scaffolding algorithm is based on the observation that similar analogical coherence can be found in sentence-level embedding spaces, albeit not as precisely or compactly structured. It is not, generally speaking, possible to mathematically compute “I like apples” - “because they are red” + “I like pears” and end up at the embedding for “because they are green”; but when one does the math, one ends up in the *general vicinity* of pears and green-ness.

Our key insight is the idea that within the context of response prioritization, *general vicinity*<sup>2</sup> is sufficient to identify candidate responses that are viable and conversationally coherent with respect to each user utterance. The basic principle is simple: A pair utterances representing the two most recent dialog turns is converted into vector representations using Google’s Universal Sentence Encoder [17]. These vectors are then matched against a conversational dataset in order to identify likely subsequent sentences. (In theory, this task could also be attempted using a sequence-to-sequence model. However, the scarcity of high-quality training data combined with the variational nature of conversation makes this a challenging proposition. Hence our decision to use a scaffolding algorithm.)

Our conversational scaffolding methods require a high-quality conversational dataset that exemplifies the type of discourse the bot should emulate. As it turns out, these are hard to come by. See the next subsection for further details.

### 7.5.2 The Chit-Chat Dataset

In order to find a suitable dataset for our scaffolding technique, we first examined the Cornell Movie-Dialogs Corpus [26] and related datasets [28, 112]. Unfortunately, movie scripts proved

---

<sup>2</sup>Here, *general vicinity* means ‘within a threshold distance’. We used a euclidean distance metric with  $.6 * \|\mathbf{h}\|$  as the threshold, where  $\|\mathbf{h}\|$  is the number of utterances in the conversation history window.

unsuitable for our purposes because of two key drawbacks. (1) The text and subject matter tend to be overly dramatic rather than reflecting normal day-to-day dialog, and (2) The characters often speak in response to physical events within the scene; thus the dialog when taken out of context becomes meaningless. Social media dialogues like Twitter [129] or online forums like Reddit [122] avoid the key pitfalls of movie scripts, but also tend to contain highly controversial material, personal insults, and incendiary opinions: not exactly the behaviors we wish our socialbot to emulate.

After reviewing and eventually rejecting the available turn-based datasets, we decided to construct our own. We outsourced our data collection methods to students at our university in the form of a competition [168] in which students were randomly paired with each other in an online chat forum and asked to discuss one of several pre-defined topics. To heighten interest and engagement we offered various prizes, and these prizes were awarded to users with the highest number of quality interactions (quality being defined as a complex relationship between post length, word length, and other factors).

The resulting Chit-Chat dataset contains over 90,000 utterances from almost 1200 different users. The data is almost entirely free of offensive or derogatory statements, however it is a little noisy; we set out to capture authentic examples of human conversation, but the task itself became a topic of conversation. Chatters tended to discuss the competition's prizes, the scoreboard, and their desire to win, which subverts the actual goal of the task and creates the possibility of strange dialogue patterns in any system that uses it. (e.g. imagine a socialbot talking about a competition to provide data for a socialbot). Overall though, the data generated from our simple competition is quite good, and this dataset has become a solid training basis for multiple components in the system architecture.

Our conversational dataset is unique in that, unlike Reddit and other popular conversational datasets, there is a minimum of personal attacks and mean-spirited discussion. During account creation, each user committed to maintain high standards of behavior and agreed to let us use and share their conversation data for research purposes. We will publicly

release the dataset at a later date, pending further stages of our Chit-Chat competition. (For example, we may want to increase the size of the dataset and/or filter out references to the contest, scoreboard, and prizes).

### 7.5.3 Scaffolding Algorithms

We are actively exploring three algorithms for response prioritization. All three methods rely on the Chit-Chat dataset to provide conversational examples: however, the method for selecting a follow-on sentence varies. We compare the performance of all three algorithms to a multi-layer network trained to predict the correct subsequent sentence embedding for a conversation history of 2.

For simplicity, the algorithm descriptions below assume a conversation history of 1, meaning the algorithm is provided only the user’s most recent utterance. However, we have found the latter three algorithms more effective when given a conversation history of 2, consisting of the user’s utterance *and* the most recent utterance provided by the socialbot. This allows the algorithm to contextualize and respond meaningfully to information-sparse utterances like “yes” or “I’m not sure”.

All three algorithms assume the availability of a set of candidate responses  $g_i$ , which represent the outputs of the socialbot’s various response generators for user utterance  $c$ .

**Naive Offset** This scaffolding algorithm is based on the simplifying assumption that the closest match for the user’s utterance within the Chit-Chat dataset is always paired with an optimal response. (In reality, Chit-Chat utterances with a slightly larger distance from the user utterance are often paired with superior responses; this is addressed in the scattershot and flow vector methods, below.)

**Scattershot** The key insight in the scattershot approach is the idea that there isn’t really a ‘single right answer’ when it comes to utterance/response pairs. Instead, there exists a manifold (or perhaps multiple manifolds) of correct responses, any of which is equally valid.

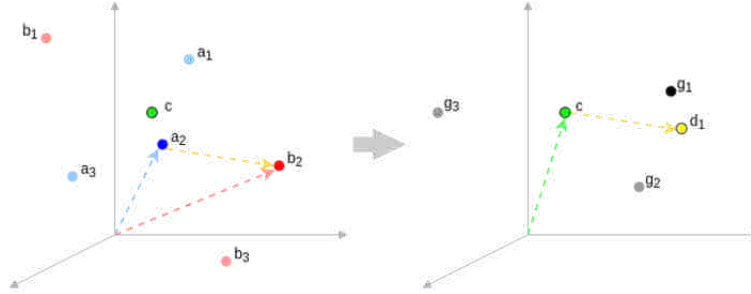


Figure 7.3: Naive Offset, where:  $c$  (green) represents the embedded utterance,  $a_i$  (blue) represent the nearest embedded utterances from the Chit-Chat Dataset,  $b_i$  (red) represent the associated embedded response to  $a_i$  in the Chit-Chat Dataset,  $d_1$  (yellow) represents the ‘ideal’ response, and  $g_i$  (grey and black) represent embedded responses generated by our system with  $g_1$  (black) representing the response selected by the Naive ranking strategy.

---

### Algorithm 7 Naive offset

---

**Inputs:**

$\vec{c}$  = Embedded user utterance

$\vec{r}$  = Candidate responses produced by the generators.

$C$  = Chit – Chat dataset

**Output:**

$S = \{s_1 \dots s_i \mid s \in [0, 1]\}$  where  $s_i$  is the score for  $r_i$

$\vec{a} \leftarrow \min(\text{dist}(\vec{c}, C))$  Where  $\text{dist}$  is any valid distance metric.  $\vec{b} \leftarrow$  Find the utterance in  $C$  that directly follows  $\vec{a}$  We ensure  $\vec{b}$  is not an end of conversation token.  $\vec{d} \leftarrow \min(\text{dist}(\vec{b}, \vec{r})) 1.0 - \frac{\vec{d}}{\|\vec{d}\|}$

---

For example, the question “Have you seen Black Panther?” Can be answered with any of [‘Yes’, ‘No’, ‘I’m going to watch it tonight’, ‘Are you kidding? Of course I’ve seen Black Panther’]. From a strictly text-based perspective, all responses are equally valid. (You’d need a knowledge graph/self-model to determine which of the valid responses are also truthful.) Accordingly, the scattershot algorithm assumes that as long as you’re close to *any one* of the valid responses, you’ve probably got a pretty good candidate.



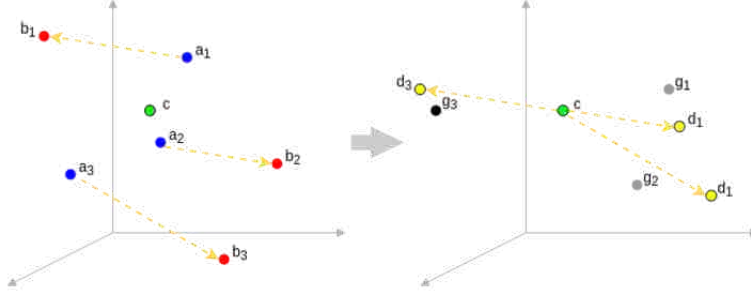


Figure 7.4: Scattershot Method, where:  $c$  (green) represents the embedded utterance,  $a_i$  (blue) represent the nearest embedded utterances from the Chit-Chat Dataset,  $b_i$  (red) represent the associated embedded response to  $a_i$  in the Chit-Chat Dataset,  $d_i$  (yellow) represent the ‘ideal’ responses, and  $g_i$  (grey and black) represent embedded responses generated by our system with  $g_3$  (black) representing the response selected by the Scattershot ranking strategy.

---

### Algorithm 8 Scattershot

---

**Inputs:**

$\vec{h} = \text{Embedded conversation history.}$

$\vec{r} = \text{Embedded candidate responses produced by the generators.}$

$n \times \|h\| - 1C = \text{Embedded Chit - Chat dataset, where the columns are pairwise differences between subsequent utterances}$

**Output:**

$S = \{s_1 \dots s_i \mid s \in [0, 1]\}$  where  $s_i$  is the score for  $r_i$

$\vec{c} \leftarrow [h_1 - h_0, \dots, h_n - h_{n-1}]$  in 1..5 Find the  $n$  closest points in  $C$  to  $c$ .  $\vec{a}_i \leftarrow \min_i(\text{dist}(\vec{u}, C))$  Where  $\text{dist}$  is any valid distance metric.  $\vec{b}_i \leftarrow$  Find the utterance in  $C$  that directly follows  $\vec{a}_i$   $\vec{d}_i \leftarrow \vec{b}_i - \vec{a}_i + \vec{c}$  Where  $d_i$  is the “ideal” response vector to  $\vec{b}_i$ .  $r_i$  in  $\vec{r}$   $\vec{g}_i \leftarrow \min(\text{dist}(\vec{r}_i, \vec{d})) 1.0 - \frac{\vec{g}}{\|\vec{g}\|}$

---

**Flow Vectors** The flow vectors approach is based on the idea that conversations tend to “flow” from certain regions of embedding space into others, and that all matching utterance pairs will reflect the same general flow direction. Rather than using a single conversation vector, this method averages multiple vectors and looks for a candidate utterance that matches the resulting flow direction.

Tables 7.1 and 7.2 shows examples of two closely related queries and a set of human-generated candidate responses. Notice that the two examples accurately distinguish the idea of being “from” or growing up somewhere from the idea of simply being born somewhere. In table 7.1, candidates that capture the idea of growing up somewhere, like “I am from California” or “I grew up in California”, are ranked higher (lower distance).

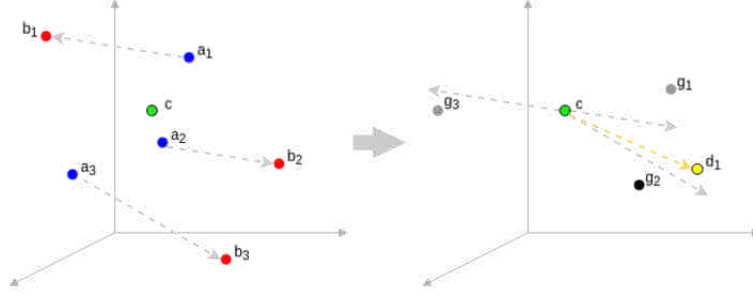


Figure 7.5: Flow Vectors Method, where:  $c$  (green) represents the embedded utterance,  $a_i$  (blue) represent the nearest embedded utterances from the Chit-Chat Dataset,  $b_i$  (red) represent the associated embedded response to  $a_i$  in the Chit-Chat Dataset,  $d_1$  (yellow) represents the ‘ideal’ response, and  $g_i$  (grey and black) represent embedded responses generated by our system with  $g_2$  (black) representing the response selected by the Flow Vectors ranking strategy.

---

### Algorithm 9 Flow vectors

---

**Inputs:**

$\vec{h} = \text{Embedded conversation history.}$

$\vec{r} = \text{Embedded candidate responses produced by the generators.}$

$n \times \|h\| - 1C = \text{Embedded Chit - Chat dataset, where the columns are pairwise differences between subsequent utterances}$

**Output:**

$S = \{s_1 \dots s_i \mid s \in [0, 1]\}$  where  $s_i$  is the score for  $r_i$

$\vec{c} \leftarrow [h_1 - h_0, \dots, h_n - h_{n-1}]$  **1..5** Find the  $n$  closest points in  $C$  to  $\vec{c}$ .  $\vec{a}_i \leftarrow \min_i(\text{dist}(\vec{c}, C))$  Where  $\text{dist}$  is any valid distance metric. We used Euclidean distance.  $\vec{b}_i \leftarrow$  Find the utterance in  $C$  that directly follows  $\vec{a}_i$ .  $\vec{F} = 1/n \sum_{i=1}^n b_i - a_i$   
 $\vec{d} = \vec{a} + \vec{F}$  Where  $\vec{d}$  is the “ideal” response vector to  $\vec{c}$ .  $\vec{g} \leftarrow \min(\text{dist}(\vec{r}, \vec{d}))$  Where  $\text{dist}$  is any valid distance metric. We used Euclidean distance.  $1.0 - \frac{\vec{g}}{\|\vec{g}\|}$

---

**Neural Network** As a baseline comparison, we implemented a multi-layer regression network using Tensorflow [1]. Its input is two utterances from the Chit-Chat dataset (see Section 7.5.2), each embedded as a 512 dimensional vector using the Universal Sentence Encoder [17]. These are concatenated to produce a 1024 dimensional input vector. The output is a single 512 dimensional vector prediction of the best response. We used two hidden layers with 2048 and 2014 units respectively, with exponential linear unit (ELU) activation functions and a Mean Squared Error (MSE) loss function. We found that training the network using a simple stochastic gradient descent optimizer with a learning rate of .001 worked best. The dropout value was 25%.

Candidate responses	Distance
I am from California.	0.789
California.	0.843
Cali.	0.845
I grew up in Santa Fe but my family just moved to Salt Lake City.	0.855
I grew up in California.	0.864
NY.	0.899
New York.	0.925
Manhattan.	0.998
Yes.	1.058
Have you ever been to California?	1.082
I love Star Wars.	1.112
Babies are usually born in a hospital.	1.203

Table 7.1: Query: “Where are you from?”

Candidate responses	Distance
California.	1.065
Cali.	1.106
NY.	1.131
New York.	1.132
Manhattan.	1.158
I am from California.	1.18
I grew up in California.	1.21
I grew up in Santa Fe but my family just moved to Salt Lake City.	1.231
Have you ever been to California?	1.277
Babies are usually born in a hospital.	1.375
Yes.	1.394
I love Star Wars.	1.487

Table 7.2: Query: “Where were you born?”

## Comparison

**offline testing** In order to evaluate the performance of the scaffolding algorithms in offline tests, we created a hand-annotated dataset taken from the bot’s dialog history. For each user utterance, an Alexa Prize team member selected the best candidate response by hand. Because this was a time-consuming process, we contented ourselves with a small initial evaluation set of 630 examples. We then measured the performance of each algorithm variant

Offline	Number Correct	Accuracy
Nave	333	52.8571%
Scattershot	337	53.4921%
Flow vectors	327	51.9048%
Neural network	402	63.8095%

Online	Avg rating	Conversations
Nave	3.14	177
Scattershot	3.11	180
Flow vectors	3.25	275
Neural network	2.99	214
No algorithm	2.99	204

Table 7.3: Top: Accuracy of scaffolding algorithms in offline tests. To count as correct, the ideal answer must be in the top 40% of the algorithm’s preferred responses. Bottom: Average rating received during A/B testing. Tests were performed between 8/18/2018 and 8/24/2018 inclusive, with each conversation randomly assigned to use one of the scaffolding variants, the neural network predictor, or no scaffolding algorithm at all. All other system parameters were held constant.

on our evaluation set. Results are shown in Table 7.3. In offline testing, we found that the neural network was the most robust algorithm.

**online testing** When deployed in production, use of the Flow Vectors algorithm created the highest level of customer satisfaction, with an average rating of 3.25 (see Table 7.3). The disparity between these results and the offline tests suggests two things: (1) The combination of the Flow Vectors algorithm with the Chit-Chat scaffolding corpus was able to generalize more effectively than the neural network to the scope and variability of real-world human conversation. (2) The Flow Vectors method, while less accurate in offline tests, has the advantage of failing with dignity in production settings. In other words, it does not reliably select the ‘optimal’ response, but the responses it chooses instead are satisfactory to customers. Further research is required to quantify this phenomenon.

**data density** The three scaffolding algorithms implemented a failsafe mechanism when  $\vec{b}$  landed in a sparse region of the Chit-Chat corpus. If the algorithm could not find enough nearby points to determine a good  $\vec{g}$ , it returned a value of 0, indicating it could not determine

a score. The raw distances were still returned as an indication of confidence. The neural network, which did not have this failsafe mechanism, was especially problematic because it simply guessed during inference when it found itself in a sparse region. During offline testing, data sparsity errors occurred approximately 25% of the time.

## **7.6 Supporting Infrastructure**

Our response generators rely on preprocessing and keyword/entity extraction. These tasks are performed by the system infrastructure, which also handles scalability, parallel processing, response ranking, and so forth. An overview of this infrastructure is provided below.

### **7.6.1 Topic Detection and Entity/Keyword Extraction**

Topic detection is accomplished using Amazon’s topic classifier, which usually runs in 600 ms or less and provides reasonably good accuracy. Entity detection is accomplished via a combination of Spacey’s entity detection capabilities and Google’s natural language processing API. Keywords are extracted using hand-coded text analysis; for example if the word ‘about’ appears in the user utterance, the words immediately following it are assumed to be keywords. All entities and keywords are filtered to prevent the agent from processing inappropriate language. More information about the filtering process is provided in section 7.6.4.

### **7.6.2 Offensive Speech Detection**

It is vital that we prevent the socialbot from using language that could be considered inappropriate, and it would be useful to know when the users are using inappropriate language so that we can approach the conversation carefully. Amazon has a built-in offensive speech classifier, but it isn’t as fast as a homemade solution.

We started with the hand-annotated blacklist from Amazon’s sample code, which consisted of a single long list of offensive words. Language is not black and white though, and we needed gray areas; there are some topics that some users might not want to hear about,

while other users actively try to discuss them. We divided the blacklist into three different categories: ‘severe’, ‘medium’, and ‘mild’. The ‘severe’ category is for words that we never want the bot to say. This list is comprised of words that are offensive alone, without context. The ‘medium’ category is for words that usually occur in the context of offensive language. The ‘mild’ category is reserved for words that indicate possibly offensive or inappropriate topics such as ‘murder’ and ‘guns’. We do not want our socialbot engaging with these topics without user permission, and possibly not at all. Yet for some users, ‘guns’ are a perfectly acceptable conversation topic, and we wanted our code to reflect that.

### **7.6.3 Intent Recognition**

We tested various methods for intent recognition, including neural networks and the use of distance metrics within sentence-level embedding spaces to classify utterances. In the end, we found that a hand-coded solution using a priori knowledge about conversational patterns was most effective. Our hand-coded intent recognizer is capable of triggering more than one intent at a time. For example, if the user said “oh yes that is wonderful”, the intent recognizer would identify “oh” as “useless”, “yes” as “yes response”, and “that is wonderful” as “positive”. This is accomplished using simple regular expressions, so there is nothing particular novel being done. That said, it works remarkably well, provided most cases of user speech can be anticipated by the system’s designers.

Future work on intent recognition could include the use of embedding grammars to generalize from a small set of hand-crafted expressions to a wide array of synonyms and related terms [170]. Preliminary experiments suggest that this would allow the system to extract intents from utterances that would otherwise remain unlabeled.

### **7.6.4 Response Evaluation (Ranking & Filtering)**

In a system with many generators, a “best response” must be chosen from among the generated candidates. Combined with the current state of the conversation, all generated

response sentences must be evaluated for consistency and relevance. This evaluation should also take into account user personality and the system’s synthetic personality elements.

During the design process, we decided that the function that selects the final response to be spoken by the system (hereafter known as the Arbiter) should accept as input a matrix containing ranked features about each possible output response. This function would evaluate the features of each sentence, rank the sentences, and return a concatenation of several highly-rated responses.

### Response Ranker List

1. **Confidence:** Each response generator hands the ranker a confidence rating, which is a measure of the value the generator places on its generated response. This requires a division of labor, as well as some evaluative code in each generator.
2. **Context:** Some response generators are better equipped to handle certain topics than others. The context ranker examines the topic of the user utterance to determine how well-equipped each generator is to address the current conversation state. For example, when the current topic is “movies”, the movie response generator will receive a higher score.
3. **Embedding Distance:** A conversational scaffolding method (see Section 7.5.1) is used to evaluate how well each candidate response matches the “flow” of the current conversation. Along with the Context and Preference rankers, this is one of the system’s most heavily-weighted factors, as shown in Figure 7.8.
4. **Intent:** This is a measure of how well a response matches the intent triggered by the user’s input sentence. The intents are shown in Figure 7.6.

Once the intents have been recognized in the user input, the intent recognizer is run on each of the generated responses, and we use the comparison between the intents found in those sentences to identify whether the responses have a compatible intent (see Figure 7.7). As an example, if the user input is found to contain the “apathetic” intent,

Intent	Brief Description
yes response	affirmations
no response	negations
imperative	command words
opinion request	question specifically about opinions
general fact request	question regarding general knowledge
name info	giving us their name
user fact	offering information about themselves
song request	sing a song
rap request	rap
meta request	question about the conversation
hard stop	stop talking right now
soft stop	say goodbye and end the conversation
opening request	general small talk
apathetic	apathetic response
emotional cause (and effects)	user describes something
emotional effect (and causes)	user describes a state he's in

Figure 7.6: A list of the created intents and a brief description of each one.

the generated response is said to match if it contains the “opinion request”, “general fact request”, or “user fact” intent. If a user is not currently engaged, the system will try to engage them with facts or questions.

5. **Keyword Model:** Measuring the state of a human’s personality is difficult, but we can measure certain aspects of their personality and try to tailor our responses to their mood and interests. This would cause EVE to have more personalized conversations with individuals, making for a more interesting experience. The keyword model tries to detect the current mood/emotion of the user, and use that to upvote certain response generators. If it detects that the user ‘wants to learn’, fact-based response generators will score more highly.
6. **Length:** The length of the response is critical in influencing user engagement. In early user trials, we observed that if the system talks for too long at a time, the user becomes less engaged with the socialbot. But this also works in reverse; if the socialbot says too little, the user also disengages. After watching several users interact with our system in real-time, we calibrated the length ranker to rank a given sentence as ideal when it’s



User Intent	Response Intent	Match
opinion request	user fact	yes
name info	opening request	yes
user fact	positive, opinion request, general fact request, meta request	yes
soft closing request	soft closing request	yes
opening request	opening request	yes
apathetic	opinion request, general fact request, user fact	yes
emotional cause	emotional effect	yes
emotional effect	emotional cause	yes
imperative	yes response, no response, opinion request, general fact request	no

Figure 7.7: A list of user intents and response intents paired to indicate either a positive or negative match.

between seven and 15 words. This is under the assumption that we will be concatenating multiple responses to create a final output with a target length between 15 and 25 words. Our socialbot will never return a response comprised of more than 30 words.

7. **Preference:** As the system’s developers, we trust certain response generators to generally answer better than others. We added the preference ranker to give individual preference to the most sophisticated and well-coded generators. Each generator is assigned a preference rank between zero and one, and that is handed to the Arbiter as a feature and is factored into the final decision.
8. **Repeat:** No one wants to hear the same sentences over and over, and no human communicates that way. The repetition ranker flags sentences that have been offered to the user in the last 15 conversation steps. This feature is essential to ensure a good user experience with each new response from the socialbot.
9. **Sentiment:** The sentiment ranker seeks to emphasize positive responses while still being respectful of the user’s mood. We use Google’s sentiment classification API to evaluate the sentiment of each sentence.

Feature	Scalar
Confidence	0.5
Context	3.0
Embedding	2.0
Distance	1.0
Intent	1.0
Keyword Model	1.0
Length	0.5
Preference	4.0
Repetition	2.0
Sentiment	0.5

Figure 7.8: A list of the features and their respective scalars.

The Arbiter’s task is to accept a feature matrix comprised of scores from the various rankers. Each feature in the matrix has an assigned scalar which is used to temper or enhance its influence in deciding the final ranking of the responses. The final ranking is the sum of the Hadamard product of the features and their respective scalars, multiplied by a throttle factor composed of the score from the throttling ranker (repetition) and its respective scalar:

$$(\textit{repetition\_ranking} * 2.0) * \sum_{i=1}^9 \textit{feature\_vector}_i * \textit{scalars}_i$$

It’s worth noting that this equation shows a feature vector (as opposed to a matrix), because here we’re showing the Arbiter running on only a single response for clarity. It’s also important to note that the ‘feature\_vector’ referenced in the equation doesn’t contain the throttling feature.

### 7.6.5 Response Combination

Once the Arbiter has produced a ranking for each candidate response, the responses are combined into an emote/answer/offer framework as described in Section 7.1.4. This is accomplished by using a sequence of hand-coded rules including the following:

1. The highest-ranked answer response is always used.

2. If an emotive response with a sufficiently high ranking exists, prepend it to the answer.
3. If the answer response does not end in a question mark or other conversational prompt, and if an offer response with a sufficiently high ranking exists, append the highest-ranked offer to the answer.

Additional rules ensure that the final response is not too long and that system does not combine two responses which are substantially similar in content. Certain response generators, such as MovieBot, generate responses that always include both an answer and an offer; no offers are ever appended to responses from these generators. Additionally, conversational ‘stitching’ is sometimes performed by removing the prefixes [‘Okay’, ‘Sure!’, ‘So’] from answer responses when an emote is prepended.

## **7.7 Future Work**

### **7.7.1 Improved Knowledge Graph Structure and Traversal**

Moving forward, we wish to expand the number and types of edges detected by DataBot and utilized by ModelqueryBot. In addition to ‘called’, ‘likes’, and ‘dislikes’ edges, we aim to explore methods that might automatically identify and curate a list of important edges to include in the entity models.

Our team’s near-field work is also focused on expanding the system functionality to model other entities besides the user and the system itself. As an example, if the user says “I really liked (movie title) because (name of actor) was hilarious.”, it would be fairly straightforward to generate a user-specific node for ‘(name of actor)’ and record that the actor is funny. The system would only remember this ‘fact’ when interacting with the specific user who provided the information, and if the system interacted at a later point in time with a user who disliked the same actor the system would adapt its knowledge graph model of that actor to reflect the negative sentiment of the new user. In this way, the system could

remember to avoid topics that the current user doesn't enjoy and direct the conversation towards topics the user finds more appealing.

By discussing particular topics with the system, the user will be able to mold the personality of the system to themselves; future work aims to enable the system itself to alter its own personality dynamically by incorporating probabilistic modeling into the user and self models. Using statistics on the models gleaned across thousands of interactions with users, probabilistic modeling could enable the system to infer unspoken opinions users might hold and the types of utterances the system should produce given the current state of its models.

### **7.7.2 Optimized Embeddings Module**

While our conversational scaffolding techniques show valuable potential, much work remains in order to determine the optimal algorithm for prioritizing candidate responses. We are in the process of evaluating the effect of scaffolding algorithm, distance thresholds, and conversation history on classification accuracy in offline tests.

### **7.7.3 Similarity Metrics for Data Retrieval**

Scaffolding is only one of many possible uses for universal sentence embeddings. In the future, we plan to deploy a data retrieval mechanism that will allow response generators to customize their information to specific user utterances. This can be done by measuring the distance (in embedding space) between the user utterance and each candidate response. For example, if the user says “Tell me news about 12 people trapped in a mine”, a nearest-neighbor search in embedding space would likely return headlines such as “Twelve boys trapped in a Thai Cave, Crews working to rescue them”, a trending news topic from July 2018. Preliminary experiments suggest that this method is more effective than a keyword search (which would have matched only the words ‘trapped in a’).

A similar approach can be used to improve the ability of response generators to correctly identify which knowledge graph node was invoked by the user's utterance. As

described in Section 7.2.2, each node in the knowledge graph is accompanied by a universal sentence embedding corresponding to the node’s description. This embedding can be used to disambiguate between identically-labeled nodes.

For example, if the user says “I watched Harry Potter this weekend”, it can be difficult for our knowledge graph algorithms to determine whether the user is referring to the “Harry Potter” node that describes a series of books, the “Harry Potter” node that describes a movie series, or the “Harry Potter” node that describes the main character in both. Selecting the wrong node can result in jarring conversational disconnects. However, since previous utterances in the conversation were likely focused on movies (and because the current utterance utilizes movie-related phrases like ‘watched’ and ‘weekend’) the average distance of the preceding sentences to the embedded description of each knowledge graph node will likely reveal the correct interpretation of the user’s meaning.

#### **7.7.4 ScriptDog: A Language for Managing Conversational State**

One challenge inherent in dialog management is tracking conversational state. As part of the competition, we developed a language specifically built to describe complex, factored dialog scripts with a natural, python-like syntax. The language is called “ScriptDog”, for “scripted dialog” [169].

The primary feature of the language is the ability to store the program state in a JSON-serializable object. This makes it easy to, for example, stash the program state in persistent storage and reload it at a later time. (This is particularly useful when program execution is broken up across multiple invocations that are distributed across multiple machines). The model makes it easy to integrate with AWS Lambda and other serverless computing frameworks. Tight integration with python makes it possible to cleanly separate conversational state from the backend logic that drives transitions, such as integration with databases, NLP processors, knowledge graphs, etc.

Other key language features include (1) factored, reusable state sequences; (2) factored, reusable transition definitions; (3) direct language support for random choices; (4) tight integration with regular expressions; (5) global transitions that are defined once, but implicitly accessible at any point. Future response generators for EVE will use the ScriptDog language and its associated features.

## 7.8 Conclusion

BYU-EVE is an open domain dialogue architecture founded on the principle that centralized knowledge representation coupled with self- and user-modeling can produce dynamic, human-like conversations. The key innovations in this research are twofold: (a) we treat conversation as a multi-channel information stream along which not only information, but also emotive and psychological data is being passed, and (b) we combine structured knowledge graph traversal with deep learning in order to produce fluid, personable responses that are grounded in known facts.

As part of Amazon’s Alexa Prize Challenge, EVE remains a work in progress. Our next research steps include improved knowledge graph traversal, optimized methods for conversational scaffolding, and dynamic text generation via neural networks conditioned on knowledge graph nodes. Our goal of achieving truly open domain, free-form conversation is ambitious, but it also presents intriguing possibilities. If it can be achieved, it will open doors for general purpose, responsive conversational systems that require far less retrieval and far less hand-coding than the present state-of-the-art. The resulting systems would positively impact fields including education, personal assistants, elder care, and many others.

## Part IV

### A Well-formulated Embedding Space

If we wish to extend analogical reasoning beyond elementary word vector calculations, we need a new kind of sentence-level embedding space. In particular, sentence-level embeddings should be viewed not merely as useful input features for machine learning algorithms, but rather as valuable common-sense repositories in their own right - knowledge-bases that can be queried mathematically, and which are able to facilitate a wide variety of tasks with little to no subsequent training.

As a consequence of our work in Parts II and III of this document, we concluded that three properties are necessary in order to create an embedding space that functions optimally at sentence-level linguistic reasoning tasks:

1. *Analogical coherence.* The analogical properties observed in word2vec, GLoVE, FastText, and other single-word embedding spaces should be preserved in the trained sentence-level vector space such that offset relationships like Spain:Madrid::France:Paris are preserved. At the phrase or sentence level, this should extend to relationships like ‘if you drop a ball’:‘it will bounce’::‘if you drop a glass’:‘it will break’.
2. *Semantic alignment.* Single-word embeddings should be located closer to sentences that contain those words than to conceptually equivalent sentences that do not contain them. Sentences that express similar ideas should be located near one another despite

variations in syntax or structural complexity, and arbitrarily-ordered “bags of words” should be located close to sentences in which those words appear.

3. *Polarity displacement.* A sentence and its negation should be located far from each other along at least one basis dimension of the space, and the negated sentence should be located close to non-negations that nevertheless convey the same concept. (For example, ‘The room is not empty’ should be located farther from the statement ‘The room is empty’ than it is from sentences such as ‘The room is occupied’ or ‘The room is full’.)

This section documents the implementation of these three criteria in a cohesive network architecture which we describe as deep constrained neural embeddings. In Chapter 8, we introduce a novel training mechanism that induces desired properties of the embedding space in a principled way. Rather than training on a smorgasbord of downstream tasks and hoping the resulting embedding will exhibit the desired properties, we instead apply direct constraints on the embedding layer. This constitutes a critical first step in the design, training, and utilization of embedding spaces with improved coherence and analogical structure: embedding spaces with the potential to change the landscape of conversational AI.



## Chapter 8

### That is Not What I Said: Improved Discrimination of Negated Sentences via Deep Constrained Neural Embeddings

by Nancy Fulda and David Wingate,  
to be submitted

#### Abstract

Neural linguistic embedding models do not always reflect semantic changes caused by small lexical differences (for example, by the inclusion of the word ‘not’). This leads to problematic behaviors when developers use sentence embeddings as an ad-hoc approximation for semantic similarity, and may have downstream ramifications when pre-trained embeddings are used for transfer learning. In this work, we present a deep recurrent neural model that is able to capture these lexical differences in the geometry of the resultant embedding space while still performing competitively at traditional tasks such as SemEval 2013. Our model is lightweight, trains quickly, and outperforms state-of-the-art models at a new semantic discernment task designed to measure how well sentence negations and antonym substitutions can be detected via simple cosine distances between embedded sentences.

#### 8.1 Introduction

In natural language understanding, small lexical changes can lead to large semantic differences, and this behavior is not reliably captured by state-of-the-art embedding models like BERT [30], Google’s Universal Encoder [16], or GPT-2 [119]. This is not particularly surprising:

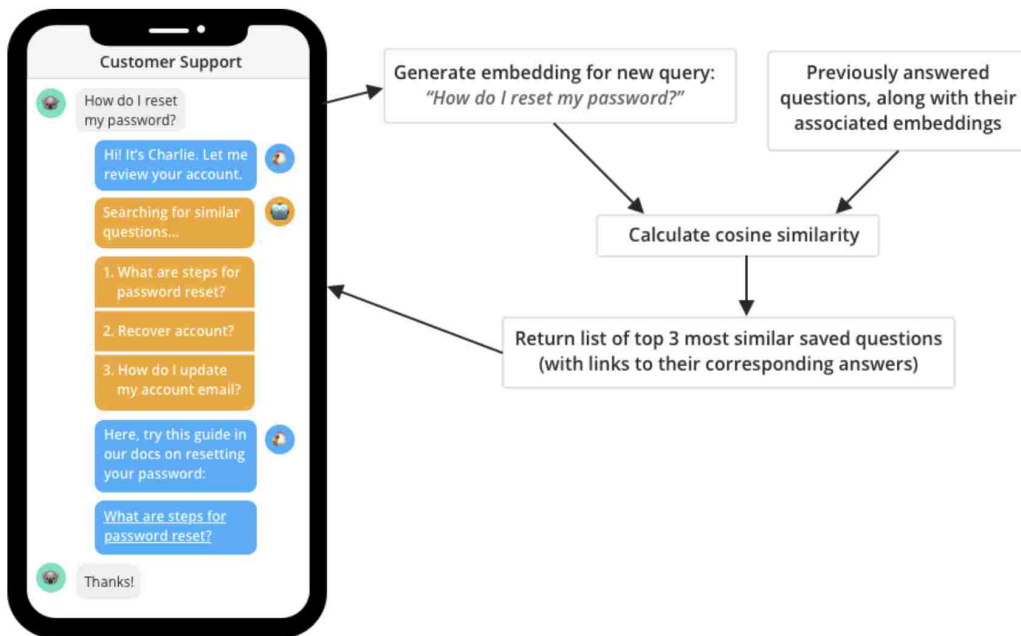
Such models were designed, not to encode semantic meaning, but to learn useful and relevant features for downstream tasks such as machine translation, natural language understanding, dialog modeling, document indexing, text classification, question answering and sentiment analysis. Semantic structure has tended to emerge as a natural byproduct of this effort rather than as the primary research objective.

However, as the popularity of neural embedding models has grown, so too has the general public notion that pre-trained embeddings can be used as a reliable heuristic for semantic meaning, with cosine distance between embedded sentences serving as a measurement of semantic similarity. This view can be found on Stack Overflow [150, 151], StackExchange [152–154], popular blog posts [63, 115, 126], and in the recommended uses of the python spacy package’s vector representations [140]. This has significant ramifications when developers attempt to use embeddings for semantic association tasks like the one depicted in Figure 8.1.

The problem, of course, is that most embedding models do not encode semantic meaning so much as semantic *relatedness*. It has long been known that vector space models trained based on context (such as skipthoughts [75] and word2vec [97]) tend to place antonym pairs in close proximity to one another [33, 43, 69, 146]. Similarly, multiword embeddings seldom distinguish adequately between contextually similar but semantically divergent sentence pairs [114, 172]. A particularly common failure mode occurs when a sentence is negated, as in the tuple {‘I want to delete my account’, ‘I don’t want to delete my account’}.

This research poses the question: Why *shouldn’t* pre-trained embeddings behave in such ways? Can we train a linguistic embedding model with the explicit goal of facilitating semantic evaluations via cosine similarity? In other words, can we create an embedding space that actually has the properties developers assume that it has?

Such a model would need to be lightweight, have a relatively small embedding size (to enable swift nearest-neighbor matching), and distinguish effectively between sentences and their negations, while still retaining the general semantic properties observed in current models. The goal is not merely to produce semantic similarity metrics, but to induce a coherent



Introducing your new AI-assisted customer support workflow

Figure 8.1: Screen Shot from FloydHub.com [63], outlining an automated customer support algorithm that relies heavily on the semantic structure of a pre-trained embedding space. The proposed algorithm is apt to deliver wildly inappropriate responses whenever the user's utterance is semantically distinct from - but contextually similar to - one of its previously-saved questions.

embedding space in which related sentences are grouped near one another in meaningful ways.

As a first step toward this goal, we present two new semantic discernment tasks designed to detect sensitivity to the lexical discrepancies described above. We then train a deep recurrent neural model on an unsupervised language task while exerting explicit constraints on the properties of the emergent embedding space. In other words, rather than training on a smorgasbord of tasks and hoping that the resultant embeddings will have desirable properties, we instead use distances between paired embeddings to induce the desired properties in a principled way.

A key contribution of our work is the addition of a unique unsupervised task in addition to the more common training task of context prediction. Reminiscent of the word2vec training algorithm, but distinct in the use of firm sentence boundaries and a variable-size context window, this training task uses the embedded representation of a central word to predict earlier and later words within the same sentence, thus anchoring the learned representations to the semantic structure of their composite words.

Our best-performing model achieves a 28% performance increase over Google’s universal sentence encoder and a 32% increase over skip-thought vectors at a negations discernment task, while remaining competitive the SemEval 2013 benchmark. It is relatively lightweight, with only 4 layers and approximately 31 million trainable weights. The output embedding is a 300-dimensional vector.

## 8.2 Related Work

As a research topic, neural embedding models are still in their infancy. The release of skip-thought vectors in 2015 [75] has been rapidly followed by more complex models including Google’s Universal Sentence Encoder [16], which features two variants: A lightweight implementation that disregards syntax in favor of a quickly trainable bag-of-words representation [65] and a large model based on a Transformer architecture structured around attention

mechanisms [157]. Both models were trained on a combination of tasks including context prediction, an input-response task, and classification tasks using supervised data such as the Stanford Natural Language Inference dataset. The recently-released BERT architecture [30] utilizes a multi-layer bidirectional Transformer encoder to create general purpose embeddings that generalize to a variety of downstream tasks. Finally, many researchers have observed that a simple average of word vectors, possibly weighted, and sometimes with PCA/SVD manipulation [165], [5] often outperform RNN and LSTM models for multi-word embeddings.

Despite this rapid progress and the impressive performance demonstrated by some of these models on downstream tasks, the research community is still only beginning to understand the behaviors and representational power of differently trained embedding spaces. Numerous researchers have observed that the geometry of learned embedding spaces is often inconsistent with human interpretations of semantic similarity [33, 43, 69, 114, 146, 172], and a key focus area of recent research is the analysis and improvement of semantic properties within such embedding spaces.

To that end, [24] and [172] have presented benchmark tasks that can be used to compare the semantic properties of embedding models, and many researchers have proposed architectural or curricular modifications in the pursuit of improved semantic representations [22, 90, 142]. Our work is closely related to [114], [43] and [148] in that we use supervised constraints to adjust the semantic structure learned via an unsupervised language task. However, the constraints we apply are different than theirs, as are the unsupervised learning tasks used as a basis.

### **8.3 A Deep Constrained Neural Network**

Our network architecture and training curriculum are built on the premise of working backward from the desired result. We ask, "What properties do we desire to induce in our model's embedding space?"

Typically, when researchers wish to develop a neural embedding model with specific properties, they do so by selecting training tasks related to those properties. But this method is only partially effective. After all, a model trained to *predict* semantic similarity is not the same as a model whose sentence representations *embody* semantic similarity.

To facilitate strong semantic structure in general and the discrimination of negated sentences in particular, we begin with two unsupervised language tasks, hereafter referred to as our *basis* tasks, that define a set of possible optimal combinations of network weights. From the perspective of the basis tasks, all of these weight configurations are equally desirable, but from a human perspective some are more preferable due to their crisp representation of semantic structure. We therefore introduce a *constraint* on the system by requiring the vector representations of specific sentence pairs to be orthogonal. Specifically, we require that the network minimizes the value  $abs(1-dist_{cos}(a, b))$  where  $a$  is the vector representation of the first sentence,  $b$  is the vector representation of the second sentence, and  $dist_{cos}$  is the cosine distance between them.

Our training curriculum relies on two basis tasks and two constraints:

1. **w2v (basis task):** Given a single input word, the model must predict the most likely neighbors of that word within the input sequence. This is reminiscent of the skip-gram task used by Mikolov et al. [97], but note that the definition of neighboring words is bounded by the start and end tokens of a single sentence. There is no negative sampling, and the context windows are not fixed. Instead, a randomly chosen window size of  $n \in \{2,4,6\}$  is chosen on each forward pass. Note also that, although the task's input value is a single word, it is still passed through the entire network architecture of Figure 8.2 to produce its embedded representation. The embedding is then used as the input to a fully-connected layer whose outputs contain log probabilities for each word in the model's vocabulary.
2. **context prediction (basis task):** Given a single input sentence, the model must predict (a) the previous sentence, and (b) the next sentence in the training corpus. This

is the sole training mechanism used by the skipthought model [75]. To learn this task, the embedded sentence representation is passed through two fully-connected single-layer networks, one of which predicts the embedded representation of the next sentence in the input corpus, and one of which predicts the representation of the previous sentence. The predicted representations are used as inputs for a simple recurrent word-level decoder with teacher forcing [166]. MSE loss between the output characters and the ground truth characters is backpropogated through the entire network.

3. **inv (constraint):** Inversion constraint. This constraint passes pre-selected sentence pairs through the encoder model. Paired sentences are identical except for 1-5 words which have been swapped with antonyms acquired from WordNet [102]. The loss function returns  $\text{abs}(1-\text{dist}_{\text{cos}}(a, b))$  where  $a$  is the vector representation of the first sentence and  $b$  is the vector representation of the second sentence.
4. **neg (constraint):** Negation constraint. This constraint passes pre-selected sentence pairs through the encoder model. Paired sentences are identical except for the insertion of the word ‘not’ after the first verb in the sentence. The loss function returns  $\text{abs}(1-\text{dist}_{\text{cos}}(a, b))$  where  $a$  is the vector representation of the first sentence and  $b$  is the vector representation of the second sentence.

Both basis tasks used the Toronto book corpus [174] as the source of input text. Input sentences were constrained to be no longer than either 50 or 512 characters, depending on the training curriculum used; longer inputs were ignored. When more than one basis task was used during training, both tasks were trained simultaneously, with the network loss at each time step being the sum of the losses of all basis tasks and constraint measurements.

## 8.4 Network Architecture

Our network is implemented as a standard feed-forward encoder whose averaged hidden states are then passed to a single fully-connected layer, as depicted in Figure 8.2. The pre-trained

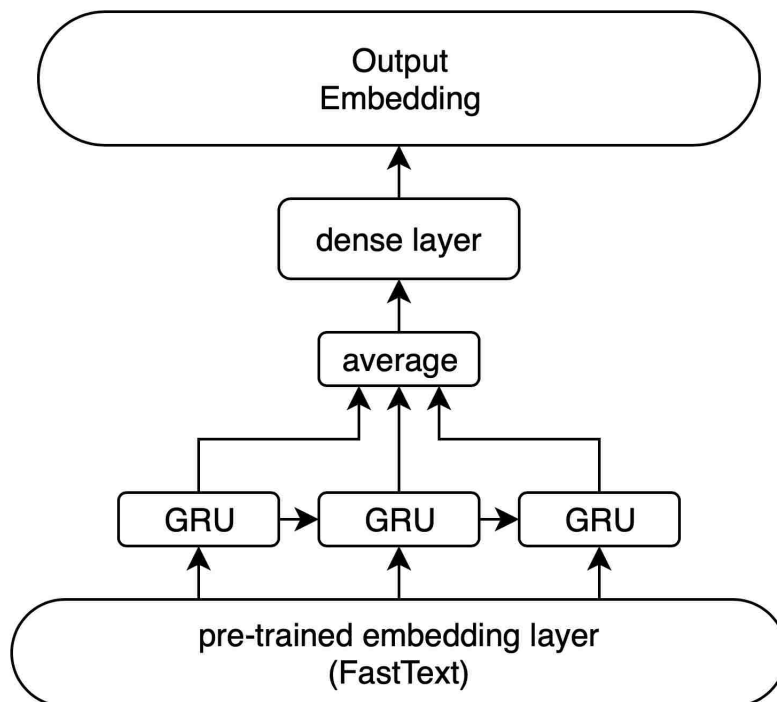


Figure 8.2: Our neural model. Pre-trained FastText embeddings are fed into a GRU encoder, with the averaged hidden states passed to a single, fully-connected layer. Although the network architecture is straightforward, the training tasks used allow it to perform comparably with state-of-the-art models on the SemEval 2013 benchmark, while outperforming them on two semantic discrimination tasks. The model is relatively lightweight, with only 4 layers and approximately 31 million trainable weights. The output embedding is a 300-dimensional vector.

embedding layer was initialized with weight values taken from FastText word embeddings [11], but was allowed to update during training.

Our model trained for two weeks on an Nvidia Tesla GPU. We used an Adam optimizer with a fixed learning rate of 0.0001, a GRU hidden size of 300, and an output embedding size of 300. FastText vectors were pruned to use only the most frequent 50,000 English words, with less common words represented by an UNK token.

## 8.5 Semantic Discernment Datasets

If an automated system is to associate sentences with one another based on cosine distances within an embedding space, it is critical that the embedding space effectively represents the



<b>Negations</b>
<i>Original:</i> this was bad.
<i>Antagonist:</i> this was not bad.
<i>Similar:</i> this was badness.
<b>Inversions</b>
<i>Original:</i> for me, that path was just beginning.
<i>Antagonist:</i> for me, that path was just ending.
<i>Similar:</i> for me, that way was just beginning.

Figure 8.3: Sample sentence trios from the *Negation* and *Inversion* evaluation sets.

semantic distinctions between antagonistic sentences. (Imagine the chaos that would ensue if an automated voice assistant were unable to distinguish between the phrases 'No, call my mother' and 'No, do *not* call my mother' or between similar but distinct commands such as 'I would like some warm milk' and 'I would like some cold milk'.

In order to measure these distinctions, we introduce the *Inversions* and *Negations* datasets. Each of these datasets consists of 10000 sentence trios: (1) An original sentence taken from the Toronto book corpus [174], (2) a semantically similar sentence constructed by swapping one or more words with close synonyms extracted from WordNet [102], and (3) an antagonistic sentence designed to be semantically distinct from the original.

In the *Inversions* dataset, the antagonistic sentence is constructed by swapping one or more words with WordNet antonyms. In the *Negations* dataset, the antagonistic sentence is constructed by inserting the word 'not' in syntactically appropriate locations. A correct evaluation of each trio occurs when  $dist_{cos}(o, s) < dist_{cos}(o, a)$ , where  $dist_{cos}$  is the cosine distance between two vectors,  $o$  is the embedded representation of the original sentence,  $s$  is the embedded representation of the semantically similar sentence, and  $a$  is the embedded representation of the antagonistic sentence.

We note that although the names are analogous, the *Inversions* and *Negations* datasets are distinct from the *inv* and *neg* constraints used during training. Sentences for both the datasets and the constraints were generated from the Toronto BookCorpus, but the constraint sentences were based off of the first half of the dataset, while the *Inversion* and *Negation* datasets were created from the second half.

In addition to these evaluation sets, we also tested each sentence-level embedding space on the SemEval 2013 dataset [167]. The purpose of this inclusion was to determine how well our embeddings perform in comparison to state-of-the-art models on an established semantic relatedness task.

## 8.6 Quantitative Analysis

Figure 8.4 shows the performance of seven well-known embedding models, as well as our deep constrained neural models, on three semantic proximity tasks. Some models, like BERT and GPT-2, were not designed to produce sentence representations *per se*, but rather to produce input features for NLP tasks. In these cases, we have done our best to extract a sentence representation from the model weights. In the case of BERT, we defined the sentence representation as the mathematical average of the contextualized word embeddings from the input sentence. For GPT-2, we defined the sentence embedding as the final hidden layer of the network, an approach that seemed to work better than averaging all of the hidden layers.

Remarkably, without the use of transformers, convolutions, attention mechanisms, or bi-directionality, four of our models are able to outperform state-of-the-art (SOTA) at either the *Inversions* or *Negations* task, and three of them have a higher average score than all SOTA models. The *Negations* task turns out to be surprisingly difficult for several SOTA models, including Google’s universal sentence encoder and the GPT-2 hidden state, both of which score below 10%. Of the pre-trained models, spacy [62] has the highest average performance and the best performance on the *Negations* task.

We attribute the success of these models primarily to the w2v basis task, which is strongly associated with high performance on the *Negations* dataset and thus raises overall average score. The context basis task, in contrast, is associated with high performance on the *Inversions* dataset. These accomplishments come at a cost, however. Most of our models show a slight reduction in accuracy on the SemEval task. We note that this reduction appears to be mitigated by the inclusion of the *inv* constraint during training. The *neg* constraint,

	<b>SemEval</b>	<b>Inv</b>	<b>Neg</b>	<b>Avg</b>
<b>SOTA models</b>				
Google use lite	38.96%	55.25%	20.12%	38.11%
Google use large	<b>41.50%</b>	52.82%	5.79%	33.37%
spacy	37.13%	56.69%	30.56%	41.46%
FastText BoW	39.80%	53.93%	4.67%	32.80%
BERT BoW	34.78%	60.7%	22.35%	39.28%
GPT-2	38.43%	49.92%	17.8%	35.38%
Skip-thought	39.94%	58.7%	16.27%	38.30%
<b>Our models</b>				
context (book corpus, len 50)	37.59%	60.38%	7.76%	35.24%
w2v (book corpus, len 50)	31.57%	54.05%	<b>58.67%</b>	<b>48.10%</b>
context+w2v (book corpus, len 50)	36.70%	60.41%	6.58%	34.56%
context+w2v (spln corpus, len 512)	37.45%	<b>61.39%</b>	19.73%	39.52%
context+w2v+inv (book corpus, len 50)	38.96%	56.92%	48.35%	48.08%
w2v+neg (wiki, len 50)	33.83%	54.52%	48.75%	45.70%

Figure 8.4: Categorization accuracy on a variety of semantic similarity and common-sense reasoning tasks. **SemEval**: A sentiment analysis task requiring the model to predict the human classification of sentences; **Inv**: An inversion task requiring discernment between two sentences in which some words have been swapped for antonyms; **Neg**: A negation task requiring discernment between two sentences that differ only by inclusion of the word ‘not’; **Avg**: The average of the values in the first three columns. The highest accuracy in each column is bolded. A box has been drawn around the model that does the best job of maximizing average score without compromising performance at SemEval.

ironically, appears to lower performance on the *Negations* dataset when combined with the already proficient w2v basis task; however, experiments not depicted here suggest that it raises proficiency at negation discernment when combined with the context task.

## 8.7 Qualitative Analysis

We next examine the behavior of each embedding model, including our w2v basis model (“our model”), on 10 sentence 4-tuples. The objective is to determine whether high performance on the *Negations* dataset is correlated with appropriate model behavior on a more human-centered task.

Figure 8.5 shows the cosine distances between pairs of sentences. In each 4-tuple, the first pair of sentences have semantically distinct meanings (e.g. ‘I am a cat’, ‘I am not a cat’) while the second pair have similar meanings (e.g. ‘I am a cat’, ‘I am a domesticated cat’). An embedding model scores a tuple correctly when the distance between semantically distinct

sentences is greater than the distance between the sentences with similar meanings. We wish to emphasize that the sentence tuples were chosen before we had evaluated our model’s performance on any of them, and that no tuples were added or removed after the initial evaluation. Thus, while these are hand-picked examples, they have not been cherry-picked.

We observe that our model and the spacy model, which have the highest percentage accuracies on the *Negations* dataset, also scored correctly on the highest number of 4-tuples. The pattern does not seem to hold at the other end of the spectrum: use-lite, which scored quite well on the *Negations* dataset, fared poorly on the sentence 4-tuples, while the FastText BoW embedding, which scored poorly on the *Negations* dataset, performed comparably with other models on the 4-tuples.

Any concrete conclusions are impeded by the small size of this qualitative analysis. As an exercise for future work, it might be profitable to compile a much larger list of 4-tuples. For now, however, the results are indicative without being completely conclusive.

## 8.8 Conclusion

What does this mean for developers, researchers, and natural language processing in general? One obvious conclusion is: "Not all sentence representations are created equal". While most of us know this in theory, it is helpful at times to see it laid out in harsh numbers. Some embedding models are best suited for use as pre-trained features for downstream tasks. Others are better at representing semantic structure.

In this paper, we have presented two datasets that might be helpful in determining which embedding spaces are best suited for which purposes. We have further developed a deep constrained neural embedding model that has been explicitly trained to distinguish between syntactically similar but semantically distinct sentences. It is our hope that this research will pave the way for future improvements in semantic discernment, and that developers will one day have neural embedding models that behave in the ways they subconsciously expect.

	use lite	use large	spacy	FastText	BERT	GPT-2	skip	our model
I am a cat								
I am not a cat	0.069	0.045	0.115	0.025	0.070	0.0007	0.069	<b>0.2047</b>
I am a cat								
I am a domesticated cat	0.098	0.111	0.150	0.072	0.188	0.0015	0.139	<b>0.0141</b>
I'm ready for this								
I'm not ready for this	0.115	<b>0.094</b>	0.091	0.014	0.091	0.0006	0.052	<b>0.0805</b>
I'm ready for this								
I am prepared for this	0.118	<b>0.066</b>	0.167	0.135	0.093	0.0017	0.128	<b>0.0266</b>
That's bad								
That's not bad	0.107	0.184	<b>0.166</b>	0.029	0.086	0.0034	0.109	<b>0.0480</b>
That's bad								
That's too bad	0.120	0.240	<b>0.155</b>	0.037	0.098	0.0134	0.119	<b>0.0009</b>
No, don't play that song again								
Play that song again	0.158	0.104	<b>0.253</b>	<b>0.178</b>	<b>0.232</b>	<b>0.0019</b>	<b>0.218</b>	0.0038
Could you play that song again								
Play that song again	0.166	0.140	<b>0.246</b>	<b>0.068</b>	<b>0.187</b>	<b>0.0009</b>	<b>0.180</b>	0.0137
Delete that file								
Don't delete that file	0.153	0.065	<b>0.366</b>	0.130	0.144	0.0007	0.217	<b>0.1045</b>
Let's keep that file								
Don't delete that file	0.379	0.273	<b>0.183</b>	0.142	0.317	0.0048	0.237	<b>0.0879</b>
I'm talking to you, Alexa								
I wasn't talking to you, Alexa	0.072	0.113	0.124	0.048	0.137	0.0019	0.073	0.0076
I'm talking to you, Alexa								
Did you hear me, Alexa	0.201	0.273	0.420	0.127	0.206	0.0029	0.158	0.0303
That movie wasn't too bad								
That movie was terrible	0.108	0.055	<b>0.320</b>	<b>0.208</b>	<b>0.158</b>	<b>0.0064</b>	<b>0.335</b>	<b>0.0269</b>
That movie wasn't too bad								
That movie was pretty good	0.113	0.065	<b>0.174</b>	<b>0.141</b>	<b>0.110</b>	<b>0.0054</b>	<b>0.311</b>	<b>0.0267</b>
I want to make a reservation								
I want to cancel my reservation	<b>0.253</b>	<b>0.330</b>	<b>0.167</b>	<b>0.090</b>	0.145	<b>0.0014</b>	<b>0.258</b>	0.0010
I want to make a reservation								
I would like to make a reservation	<b>0.084</b>	<b>0.051</b>	<b>0.084</b>	<b>0.030</b>	0.181	<b>0.0003</b>	<b>0.129</b>	0.1595
That's fair								
That's not fair	0.248	<b>0.233</b>	0.211	0.030	0.186	0.0032	0.120	<b>0.0448</b>
That's fair								
That seems fair to me	0.270	<b>0.230</b>	0.214	0.238	0.353	0.0076	0.584	<b>0.0126</b>
I was born in California								
I was not born in California	0.071	0.088	<b>0.113</b>	0.024	<b>0.081</b>	0.0015	0.087	<b>0.0036</b>
I was born in California								
I was born in San Francisco	0.120	0.146	<b>0.098</b>	0.134	<b>0.075</b>	0.0030	0.304	<b>0.0001</b>
Number of tuples correct:	1/10	3/10	6/10	3/10	3/10	3/10	3/10	7/10

Figure 8.5: cosine distances between sentences embedded using various neural models. Each sentence pair comprises two rows, with distances shown after the second component sentence. 4-tuples that evidence semantically appropriate distances are bolded.

## Part V

### The Road Forward

A wise man once told me that a dissertation is a doorway, not a destination. Like a book with ever more pages, the research documented here has produced at least as many questions as answers: How can we train more effective embedding spaces? What evaluation criteria should we use to measure their utility? What other kinds of knowledge can be harvested, encoded and extracted?

One might say that ‘The field is white already to harvest’ [136]. Looking forward, I envision a future in which linguistic embeddings are not treated as mere input features for downstream tasks, but are instead suited for use as sophisticated common-sense knowledge repositories to facilitate a wide variety of cognitive functions.

Further research in this area should focus on theoretical examinations of the semantic properties embedding spaces, ideally in conjunction with a formal analysis of linguistic embeddings as a reasoning engine. High-quality benchmark evaluation tasks should be developed to encourage collaborative efforts and to facilitate the comparison of results among researchers. Further embedding spaces should be trained, and their utility should be demonstrated on real-world tasks with human consequences. Finally, the principles explored and developed here are highly applicable to more generalized embedding spaces, especially those in which language, visual observations, and other sensor input are jointly represented in multimodal configurations.

As the work goes forward, I intend to supervise the development of truly exceptional conversational agents. This will require a fast, scalable, lightweight implementation of enriched knowledge harvested from uncurated corpora, and well-structured semantic embedding spaces show strong potential as a method for achieving this goal. Ultimately, I hope to develop embedding models that will be useful both to academia and to industry professionals, and that will help advance multiple sub-fields related to General AI.

## References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [2] H. Al-Zubaide and A. A. Issa. Ontbot: Ontology based chatbot. In *International Symposium on Innovations in Information and Communications Technology*, pages 7–12, Nov 2011. doi: 10.1109/ISIICT.2011.6149594.
- [3] Davide Alocci, Julien Mariethoz, Oliver Horlacher, Jerven T. Bolleman, Matthew P. Campbell, and Frederique Lisacek. Property graph vs rdf triple store: A comparison on glycan substructure search, December 2015. URL <https://doi.org/10.1371/journal.pone.0144578>.
- [4] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998. ISBN 9780262529204.
- [5] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.
- [6] Rafael E Banchs and Haizhou Li. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations*, pages 37–42. Association for Computational Linguistics, 2012.
- [7] Marco Baroni and Georgiana Dinu. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *In ACL*, 2014.
- [8] Alexander Bartl and Gerasimos Spanakis. A retrieval-based dialogue system utilizing utterance and context embeddings. In *16th IEEE International Conference on Machine*



- Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*, pages 1120–1125, 2017. doi: 10.1109/ICMLA.2017.00011. URL <https://doi.org/10.1109/ICMLA.2017.00011>.
- [9] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
- [10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.
- [11] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.
- [12] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *NIPS*, pages 4349–4357. Curran Associates, Inc., 2016.
- [13] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai. Quantifying and reducing stereotypes in word embeddings. *CoRR*, abs/1606.06121, 2016. URL <http://arxiv.org/abs/1606.06121>.
- [14] Kevin K Bowden, Shereen Oraby, Amita Misra, Jiaqi Wu, Stephanie Lukin, and Marilyn Walker. Data-driven dialogue systems for social agents. In *Advanced Social Interaction with Agents*, pages 53–56. Springer, 2019.
- [15] 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.
- [16] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018. URL <http://arxiv.org/abs/1803.11175>.
- [17] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018. URL <http://arxiv.org/abs/1803.11175>.

- [18] Franck Charras, Guillaume Dubuisson Duplessis, Vincent Letard, Anne-Laure Ligozat, and Sophie Rosset. Comparing system-response retrieval models for open-domain and casual conversational agent. In *Second Workshop on Chatbots and Conversational Agent Technologies (WOCHAT@ IVA2016)*, 2016.
- [19] Yanguang Chen. On the four types of weight functions for spatial contiguity matrix. *Letters in Spatial and Resource Sciences*, 5(2):65–72, 2012.
- [20] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [21] Clarifai. Clarifai image captioning demo. <https://www.clarifai.com/demo>, 2017.
- [22] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017. URL <http://arxiv.org/abs/1705.02364>.
- [23] 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.
- [24] 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.
- [25] Bo Dai, Dahua Lin, Raquel Urtasun, and Sanja Fidler. Towards diverse and natural image descriptions via a conditional gan. *arXiv preprint arXiv:1703.06029*, 2017.
- [26] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.
- [27] Souripriya Das, Jagannathan Srinivasan, Eugene Inseok, and Jayanta Benerjee. A tale of two graphs: Property graphs as rdf in oracle, 2014. URL [https://openproceedings.org/EDBT/2014/edbticdt2014industrial\\_submission\\_28.pdf](https://openproceedings.org/EDBT/2014/edbticdt2014industrial_submission_28.pdf).
- [28] The Internet Movie Script Database. The internet movie script database. URL <http://www.imsdb.com/>.

- [29] Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156, 2016.
- [30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [31] Hao Dong, Jingqing Zhang, Douglas McIlwraith, and Yike Guo. I2t2i: Learning text to image synthesis with textual data augmentation. *arXiv preprint arXiv:1703.06676*, 2017.
- [32] Cícero Nogueira dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78. ACL, 2014.
- [33] Zehao Dou, Wei Wei, and Xiaojun Wan. Improving word embeddings for antonym detection using thesauri and sentiwordnet. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 67–79. Springer, 2018.
- [34] Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. Word embeddings, analogies, and machine learning: Beyond king - man+ woman= queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, 2016.
- [35] Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530, 2016.
- [36] Guillaume Dubuisson Duplessis, Franck Charras, Vincent Letard, Anne-Laure Ligozat, and Sophie Rosset. Utterance Retrieval based on Recurrent Surface Text Patterns. In *39th European Conference on Information Retrieval*, Aberdeen, United Kingdom, April 2017. URL <https://hal.archives-ouvertes.fr/hal-01436052>.
- [37] Ahmed Fadhil. Beyond patient monitoring: Conversational agents role in telemedicine & healthcare support for home-living elderly individuals. *arXiv preprint arXiv:1803.06000*, 2018.
- [38] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

- [39] J.R. Firth. A synopsis of linguistic theory 1930-1955. pages 1–32, 1957.
- [40] Python Software Foundation. gremlinpython 3.3.3, May 2018. URL <https://pypi.org/project/gremlinpython/>.
- [41] The Apache Software Foundation. Apache tinkerpop, April 2015. URL <https://tinkerpop.apache.org/>.
- [42] Andrea Frome, Greg S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *In NIPS*, 2013.
- [43] Peng Fu, Zheng Lin, Fengcheng Yuan, Weiping Wang, and Dan Meng. Learning sentiment-specific word embedding via global sentiment representation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [44] Nancy Fulda, Ben Murdoch Daniel Ricks, and David Wingate. Informing action primitives through free-form text. In *NIPS Workshop on Visually Grounded Interaction and Language*, 2017.
- [45] Nancy Fulda, Danieal 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. doi: 10.24963/ijcai.2017/144. URL <https://doi.org/10.24963/ijcai.2017/144>.
- [46] 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. doi: 10.24963/ijcai.2017/144. URL <https://doi.org/10.24963/ijcai.2017/144>.
- [47] 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.
- [48] 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.
- [49] Nancy Fulda, Tyler Etchart, William Myers, Daniel Ricks, Zachary Brown, Joseph Szendre, Ben Murdoch, Andrew Carr, and David Wingate. Byu-eve: Mixed initiative

- dialog via structured knowledge graph traversal and conversational scaffolding. In *Proceedings of the 2018 Amazon Alexa Prize*, November 2018.
- [50] Sudeep Gandhe and David Traum. Surface text based dialogue models for virtual humans. In *Proceedings of the SIGDIAL 2013 Conference*, pages 251–260, 2013.
- [51] Dario Garcia-Gasulla, Javier Béjar, Ulises Cortés, Eduard Ayguadé, and Jesús Labarta. Extracting visual patterns from deep learning representations. *CoRR*, abs/1507.08818, 2015.
- [52] Sahar Ghannay, Benoit Favre, Yannick Esteve, and Nathalie Camelin. Word embedding evaluation and combination. In *LREC*, 2016.
- [53] James J. Gibson. The theory of affordances. In Robert Shaw and John Bransford, editors, *Perceiving, Acting, and Knowing*. 1977.
- [54] Terry Gilliam, Terry Jones, Graham Chapman, John Cleese, Eric Idle, and Michael Palin. *Monty python and the holy grail*. 1975.
- [55] 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.
- [56] Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, 2016.
- [57] Christian Grimme, Mike Preuss, Lena Adam, and Heike Trautmann. Social bots: Human-like by means of human control? *Big data*, 5(4):279–293, 2017.
- [58] Z. Harris. Distributional structure. 1954.
- [59] Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. Towards an open-domain conversational system fully based on natural language processing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 928–939, 2014.
- [60] Geoffrey E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12. Hillsdale, NJ: Erlbaum, 1986.

- [61] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [62] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.
- [63] Cathal Horan. Using sentence embeddings to automate customer support, part one. <https://blog.floydhub.com/automate-customer-support-part-one/>, December 2018.
- [64] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882. Association for Computational Linguistics, 2012.
- [65] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daume III. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*, 2015. URL [docs/2015\\_acl\\_dan.pdf](https://arxiv.org/abs/1508.07909).
- [66] Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648, 2007.
- [67] Russell Kaplan, Christopher Sauer, and Alexander Sosa. Beating atari with natural language guided reinforcement learning. *CoRR*, abs/1704.05539, 2017. URL <http://arxiv.org/abs/1704.05539>.
- [68] Andrej Karpathy, Armand Joulin, and Li Fei-fei. Deep fragment embeddings for bidirectional image sentence mapping. In *In arXiv:1406.5679*, 2014.
- [69] Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier. Adjusting word embeddings with semantic intensity orders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 62–69, 2016.
- [70] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [71] Jamie Kiros and William Chan. Inferlite: Simple universal sentence representations from natural language inference data. In *Proceedings of the 2018 Conference on*

- Empirical Methods in Natural Language Processing*, pages 4868–4874. Association for Computational Linguistics, 2018. URL <http://aclweb.org/anthology/D18-1524>.
- [72] Ryan Kiros. Sent2vec encoder and training code from the paper "skip-thought vectors". <https://github.com/ryankiros/skip-thoughts>, 2017.
- [73] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014. URL <http://arxiv.org/abs/1411.2539>.
- [74] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.
- [75] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. pages 3294–3302, 2015.
- [76] Sheldon Klein, Stephen L Lieman, and Gary Lindstrom. Diseminer: a distributional-semantic inference maker. 1966.
- [77] Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie L. Webber. Edina: Building an open domain socialbot with self-dialogues. *CoRR*, abs/1709.09816, 2017. URL <http://arxiv.org/abs/1709.09816>.
- [78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012. URL <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>.
- [79] Pavel Kucherbaev, Achilleas Psyllidis, and Alessandro Bozzon. Chatbots as conversational recommender systems in urban contexts. In *Proceedings of the International Workshop on Recommender Systems for Citizens*, page 6. ACM, 2017.
- [80] John E. Laird and Michael van Lent. Human-level AI’s killer application: Interactive computer games. *AI Magazine*, 22(2):15–26, 2001.
- [81] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

- [82] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.
- [83] 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.
- [84] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. *arXiv e-prints*, art. arXiv:1710.03957, October 2017.
- [85] Ser Ling Lim and Ong Sing Goh. Intelligent conversational bot for massive online open courses (moocs). *CoRR*, abs/1601.07065, 2016. URL <http://arxiv.org/abs/1601.07065>.
- [86] Tal Linzen. Issues in evaluating semantic spaces using word analogies. *CoRR*, abs/1606.07736, 2016. URL <http://arxiv.org/abs/1606.07736>.
- [87] H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226, 2004. ISSN 1573-1995. doi: 10.1023/B:BTTJ.0000047600.45421.6d. URL <http://dx.doi.org/10.1023/B:BTTJ.0000047600.45421.6d>.
- [88] Yang Liu, Zhiyuan Liu, Tat-seng Chua, and Maosong Sun. Topical word embeddings. In *In TwentyNinth AAAI Conference on Artificial Intelligence*, 2015.
- [89] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=rJvJXZb0W>.
- [90] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *ICML*, 2018.
- [91] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. pages 3431–3440, 2015.
- [92] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. *arXiv e-prints*, art. arXiv:1506.08909, June 2015.



- [93] Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *D&D*, 8(1):31–65, 2017. URL <http://dad.uni-bielefeld.de/index.php/dad/article/view/3698>.
- [94] Rastislav Lukac, Bogdan Smolka, Konstantinos N Plataniotis, and Anastasios N Venetianopoulos. Selection weighted vector directional filters. *Computer Vision and Image Understanding*, 94(1):140 – 167, 2004. ISSN 1077-3142. doi: <http://dx.doi.org/10.1016/j.cviu.2003.10.013>. URL <http://www.sciencedirect.com/science/article/pii/S107731420300198X>. Special Issue: Colour for Image Indexing and Retrieval.
- [95] Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An introduction to the syntax and content of *cyc*. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.
- [96] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
- [97] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
- [98] 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.
- [99] 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.
- [100] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. Association for Computational Linguistics, May 2013.
- [101] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. Association for Computational Linguistics, May 2013.
- [102] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: [10.1145/219717.219748](https://doi.org/10.1145/219717.219748). URL <http://doi.acm.org/10.1145/219717.219748>.

- [103] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL <http://dx.doi.org/10.1038/nature14236>.
- [104] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.
- [105] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Modeling affordances using bayesian networks. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4102–4107, Oct 2007. doi: 10.1109/IROS.2007.4399511.
- [106] Karthik Narasimhan, Tejas D. Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *CoRR*, abs/1506.08941, 2015. URL <http://arxiv.org/abs/1506.08941>.
- [107] Stefan Escalda Navarro, Nicolas Gorges, Heinz Wörn, Julian Schill, Tamim Asfour, and Rüdiger Dillmann. Haptic object recognition for multi-fingered robot hands. In *2012 IEEE Haptics Symposium (HAPTICS)*, pages 497–502. IEEE, 2012.
- [108] Aida Nematzadeh, Stephan C Meylan, and Thomas L Griffiths. Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words.
- [109] Aida Nematzadeh, Stephan C Meylan, and Thomas L Griffiths. Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. 2017.
- [110] Lasguido Nio, Sakriani Sakti, Graham Neubig, Koichiro Yoshino, and Satoshi Nakamura. Neural network approaches to dialog response retrieval and generation. *IEICE Transactions*, 99-D(10):2508–2517, 2016. doi: 10.1587/transinf.2016SLP0018. URL <https://doi.org/10.1587/transinf.2016SLP0018>.
- [111] Timothy J O’Donnell, Joshua B Tenenbaum, and Noah D Goodman. Fragment grammars: Exploring computation and reuse in language. 2009.

- [112] Film Scripts Online. Film scripts online. URL <http://filmscriptsonline.com/>.
- [113] Joseph C. Osborn, Adam Summerville, and Michael Mateas. Automated game design learning. *Computational Intelligence in Games*, 2017. URL <http://arxiv.org/abs/1707.03333>.
- [114] Badri N Patro, Vinod K Kurmi, Sandeep Kumar, and Vinay P Namboodiri. Learning semantic sentence embeddings using sequential pair-wise discriminator. *arXiv preprint arXiv:1806.00807*, 2018.
- [115] Yves Peirsman. Comparing sentence similarity methods. <http://nlp.town/blog/sentence-similarity/>, 2018.
- [116] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.
- [117] 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. URL <http://aclweb.org/anthology/D/D14/D14-1162.pdf>.
- [118] Bastian Quilitz and Ulf Leser. Querying distributed rdf data sources with sparql. In Sean Bechhofer, Manfred Hauswirth, Jorg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications*, pages 524–538, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-68234-9.
- [119] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.
- [120] Geetanjali Rakshit, Kevin K Bowden, Lena Reed, Amita Misra, and Marilyn Walker. Debbie, the debate bot of the future. In *Advanced Social Interaction with Agents*, pages 45–52. Springer, 2017.
- [121] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrew. Conversational AI: the science behind the alexa prize. *Alexa Prize Proceedings*, abs/1801.03604, 2018. URL <http://arxiv.org/abs/1801.03604>.

- [122] Reddit. Reddit datasets. URL <https://www.reddit.com/r/datasets/>.
- [123] Scott E. Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016. URL <http://arxiv.org/abs/1605.05396>.
- [124] Joseph Reisinger and Raymond J. Mooney. Multi-prototype vector-space models of word meaning. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2010)*, pages 109–117, 2010.
- [125] Daniel Ricks. Autoplay: a learning environment for interactive fiction. <https://github.com/danielricks/autoplay>, 2016.
- [126] Aadhil Rushdy. Word embeddings for sentence classification. <https://towardsdatascience.com/word-embeddings-for-sentence-classification-c8cb664c5029>, 2018.
- [127] Thomas A Russ, Cartic Ramakrishnan, Eduard H Hovy, Mihail Bota, and Gully APC Burns. Knowledge engineering tools for reasoning with scientific observations and interpretations: a neural connectivity use case. *BMC bioinformatics*, 12(1):351, 2011.
- [128] Tanay Kumar Saha, Shafiq R. Joty, Naeemul Hassan, and Mohammad Al Hasan. Dis2v: Discourse informed sen2vec. *CoRR*, abs/1610.08078, 2016. URL <http://arxiv.org/abs/1610.08078>.
- [129] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for twitter sentiment analysis. a survey and a new dataset, the sts-gold, 12 2013.
- [130] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.
- [131] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra Kumar Misra, and Hema Swetha Koppula. Robobrain: Large-scale knowledge engine for robots. *CoRR*, abs/1412.0691, 2014. URL <http://arxiv.org/abs/1412.0691>.
- [132] Wolfram Schenck, Hendrik Hasenbein, and Ralf Möller. Detecting affordances by mental imagery. In Alessandro G. Di Nuovo, Vivian M. de la Cruz, and Davide Marocco, editors, *Proceedings of the SAB Workshop on “Artificial Mental Imagery”*, pages 15–18, Odense (Danmark), 2012.
- [133] Wolfram Schenck, Hendrik Hasenbein, and Ralf Möller. Detecting affordances by visuomotor simulation. *arXiv preprint arXiv:1611.00274*, 2016.

- [134] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, 2016.
- [135] Heung-yeung Shum, Xiao-dong He, and Di Li. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1):10–26, 2018.
- [136] Joseph Smith. The doctrine and covenants of the church of jesus christ of latter-day saints, chapter 4, verse 4. 1834.
- [137] Richard Socher, Cliff C. Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. *ICML*, pages 129–136, 2011.
- [138] Hyun Oh Song, Mario Fritz, Chunhui Gu, and Trevor Darrell. Visual grasp affordances from appearance-based cues. In *ICCV Workshops*, pages 998–1005. IEEE, 2011. ISBN 978-1-4673-0062-9. URL <http://dblp.uni-trier.de/db/conf/iccvw/iccvw2011.html#SongFGD11>.
- [139] Hyun Oh Song, Mario Fritz, Daniel Goehring, and Trevor Darrell. Learning to detect visual grasp affordance. In *IEEE Transactions on Automation Science and Engineering (TASE)*, 2015.
- [140] spaCy. Word vectors and semantic similarity. <https://spacy.io/usage/vectors-similarity>, 2016-2019.
- [141] Alexander Stoytchev. *Learning the Affordances of Tools Using a Behavior-Grounded Approach*, pages 140–158. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-77915-5. doi: 10.1007/978-3-540-77915-5\_10. URL [http://dx.doi.org/10.1007/978-3-540-77915-5\\_10](http://dx.doi.org/10.1007/978-3-540-77915-5_10).
- [142] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.
- [143] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.
- [144] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

- [145] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. 32:566–590, 04 2013.
- [146] Bruna Thalenberg. Distinguishing antonyms from synonyms in vector space models of semantics. Technical report.
- [147] Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. Rich image captioning in the wild. June 2016. URL <https://www.microsoft.com/en-us/research/publication/rich-image-captioning-in-the-wild/>.
- [148] Eleni Triantafillou, Jamie Ryan Kiros, Raquel Urtasun, and Richard Zemel. Towards generalizable sentence embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 239–248, 2016.
- [149] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, 2010. ISSN 1076-9757.
- [150] username: alvas. Calculate cosine similarity given 2 sentence strings. <https://stackoverflow.com/questions/15173225/calculate-cosine-similarity-given-2-sentence-strings>, 2013.
- [151] username: analyticalpicasso. How to calculate the sentence similarity using word2vec model of gensim with python. <https://stackoverflow.com/questions/22129943/how-to-calculate-the-sentence-similarity-using-word2vec-model-of-gensim-with-pyt>, 2016.
- [152] username: DaveTheAl. Best practical algorithm for sentence similarity. <https://datascience.stackexchange.com/questions/25053/best-practical-algorithm-for-sentence-similarity>, 2017.
- [153] username: lte. Sentence similarity prediction. <https://datascience.stackexchange.com/questions/23969/sentence-similarity-prediction>, 2017.
- [154] username: Shubham Tiwari. Sentence similarity in python. <https://ai.stackexchange.com/questions/4965/sentence-similarity-in-python>, 2018.
- [155] Guido van Rossum et al. Cpython github, re.py, 2017. URL <https://github.com/python/cpython/blob/33dddac00ba8d9b72cf21b8698504077eb3c23ad/Lib/re.py>.

- [156] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz 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. URL <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>.
- [157] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz 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. URL <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>.
- [158] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017. URL <http://arxiv.org/abs/1708.04782>.
- [159] Denny Vrandečić and Markus Krotzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014. ISSN 0001-0782. doi: 10.1145/2629489. URL <http://doi.acm.org/10.1145/2629489>.
- [160] Markus Waibel, Michael Beetz, Raffaello D’Andrea, Rob Janssen, Moritz Tenorth, Javier Civera, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, J.M.M. Montiel, Alexander Perzylo, Björn Schießle, Oliver Zweigle, and René van de Molengraft. RoboEarth - A World Wide Web for Robots. *Robotics & Automation Magazine*, 18(2):69–82, 2011.
- [161] Zihao Wang, Ali Ahmadvand, Jason Ingyu Choi, Payam Karisani, and Eugene Agichtein. Emersonbot: Information-focused conversational ai emory university at the alexa prize 2017 challenge. *AWS*, 2017.
- [162] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

- [163] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.
- [164] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, pages 2764–2770. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8.
- [165] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.
- [166] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.
- [167] Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Alan Ritter, Sara Rosenthal, and Veselin Stoyanov. Sentiment analysis in twitter. <http://www.cs.york.ac.uk/semEval-2013/task2/>, 2013.
- [168] David Wingate. Byu 2018 chit chat challenge, April 2018. URL <https://chitchatchallenge.com/>.
- [169] David Wingate and Tyler Etchart. Scriptdog. <https://github.com/BYU-PCCL/scriptdog/>, 2018.
- [170] David Wingate, William Myers, Nancy Fulda, and Tyler Etchart. Embedding Grammars. *ArXiv e-prints*, August 2018.
- [171] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. pages 496–505, 01 2017.
- [172] 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.
- [173] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014.
- [174] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.



- [175] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724, 2015.