Commonsense Knowledge for 3D Modeling: A Machine Learning Approach

Kaveh Hassani

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree

Doctor of Electrical and Computer Engineering

uOttawa

School of Electrical Engineering and Computer Science

University of Ottawa

Ottawa, Ontario, Canada

May 2017

Abstract

Common–sense knowledge is a collection of non–expert and agreed–upon facts and information about the world shared among most people past early childhood based on their experiences. It includes uses of objects, their properties, parts and materials, their locations, spatial arrangements among them; location and duration of events; arguments, preconditions and effects of actions; urges and emotions of people, etc. In creating 3D worlds and especially text–to–scene and text–to–animation systems, this knowledge is essential to eliminate the tedious and low–level tasks, thus allowing users to focus on their creativity and imagination. We address tasks related to five categories of common–sense knowledge that is required by such systems including: (1) spatial role labeling to automatically identify and annotate a set of spatial signals within a scene description in natural language; (2) grounding spatial relations to automatically position an object in 3D world; (3) inferring spatial relations to extract symbolic spatial relation between objects to answer questions regarding a 3D world; (4) recommending objects and their relative spatial relations given a recent manipulated object to auto–complete a scene design; (5) learning physical attributes (e.g., *size*, *weight*, and *speed*) of objects and their corresponding distribution. We approach these tasks by using deep learning and probabilistic graphical models and exploit existing datasets and web content to learn the corresponding common–sense knowledge.

Acknowledgement

I would like to express my sincere appreciation to Dr. Won–Sook Lee for her constant support during my doctorate studies. Her critical comments and insightful feedbacks were essential for any steps towards the progress of my research.

Dedication

To my

Mother, father, sister, and lovely wife

For their endless love, support, and encouragement

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
AABB	Axis–Aligned Bounding Boxes
AMT	Amazon Mechanical Turk
BOW	Bag–Of–Words
CRF	Conditional Random Field
CNN	Convolutional Neural Networks
DFA	Deterministic Finite Automata
DNN	Deep Neural Networks
EM	Expectation Maximization
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
K–NN	K–Nearest Neighbors
LSTM	Long Short–Term Memory
MLP	Multi–Layer Perceptron
NER	Named Entity Recognition
PAR	Parameterized Action Representation
PCFG	Probabilistic Context–Free Grammar
POS	Part–of–Speech
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SRL	Semantic Role Labeling
SVM	Support Vector Machine
TF–IDF	Term Frequency–Inverse Document Frequency
WIMP	Windows, Icons, Menus and Pointer

Chapter 1. Introduction

1.1 Motivation

Common–sense knowledge (i.e., consensus reality) is a collection of non–expert and agreed–upon facts and information about the world shared among the majority of people past early childhood based on their experiences [1]. It includes uses of objects, their properties, parts and materials, their locations, spatial arrangements among them; location and duration of events; arguments, preconditions and effects of actions; urges and emotions of people, etc. Learning and inferring such knowledge is one of the key aspects of Artificial General Intelligence (AGI) and plays an import role in Artificial Intelligence (AI) applications [2]. These applications include but are not limited to robotics [3], conversational agents [4], text–to–scene conversion systems [5], and visual question answering [6].

Creating 3D contents and animating them in virtual worlds is a time–consuming and labor–intensive process that requires users to elaborate on many details such as scales and spatial arrangements to achieve natural–looking environments. Automating such processes is one of the ambitious goals of computer graphics community. An intelligent system developed for this task requires common–sense knowledge of physical attributes of objects, spatial arrangements, and co–occurrences of objects to free a user from tedious labor. Moreover, given the importance of spatial relations in describing 3D scenes in text–to–scene [5] and text–to–animation [7] conversion systems, and considering the sophistications of semantic analysis of input utterances containing spatial relation, a system should be able to automatically detect and label the elements of spatial relations with proper spatial roles.

Most systems follow a knowledge–based approach by hand–crafting required knowledge (e.g., object scales) and patterns (e.g., regular expressions to capture spatial relations) to achieve these goals. Considering the huge variations in object categories, spatial arrangements, and possible input utterances containing spatial signals, these systems have several drawbacks including: (1) being prone to missing knowledge; (2) requiring a tedious process of knowledge engineering and maintenance; and (3) restricted allowed input spatial language. The goal of this

study is to address these issues by developing adaptive components that automatically learn and extract the mentioned aspects of common–sense knowledge for developing text–to–scene and text–to–animation conversion systems.

1.2 Objectives

We address tasks related to five categories of common–sense knowledge that is required by such systems including: (1) spatial role labeling to automatically identify and annotate a set of spatial signals within a scene description in natural language; (2) grounding spatial relations to automatically position an object in 3D word; (3) inferring spatial relations to extract symbolic spatial relation between objects to answer questions regarding a 3D world; (4) recommending objects and their relative spatial relations given a recent manipulated object to auto–complete a scene design; (5) learning physical attributes (e.g., *size*, *weight*, and *speed*) of objects and their corresponding distributions. We approach these tasks by using deep learning and probabilistic graphical models and exploit existing datasets and web content to learn the corresponding common–sense knowledge.

1.3 Requirements and Challenges

We follow a data–driven approach to automatically extract and model the mentioned aspects of common–sense knowledge. Thus, our approach inherits the same set of requirements and challenges that other data–driven systems face. We categorize these challenges to four classes as follows.

1.3.1 Heterogeneous Data Resources

Constructing data–driven systems and learning models requires data. However, collecting a proper dataset for a given task is not a trivial task. This process faces more sophistications for tasks that are associated with aspects of common–sense knowledge. Depending on the task and availability of resources, we address this requirement as follows. For tasks with available datasets, we use the available datasets. For other tasks, we either use Amazon Mechanical Turk (AMT) crowd–sourcing platform or crawl the web content to collect batches of data. We also use

Google search API to collect data in real–time. Moreover, we develop various preprocessing components to automatically perform data extraction (e.g., retrieving data from HTML documents), data normalization (e.g., unit conversion), data formatting (e.g., XML to JSON mapping), and data cleaning (e.g., removing noise). Finally, we develop an unsupervised semantic taxonomy merging scheme to integrate data collected from different resources into a reference taxonomy.

1.3.2 Feature Design

Feeding raw data to a learning model does not result in an expected performance. This is because raw data does not contain sufficient discriminative signals. Hence, it is required to represent it as a set of discriminative features before feeding it to a model. These features can vary from semantic features to geometric features for different tasks. As an example, given a corpus of sentences, a classifier trained on words and syntactic tags will outperform a classifier trained only on words. Nevertheless, deciding a proper set of features to represent data is not a trivial task. Depending on the task, we use feature engineering (i.e., explicit feature design), representation learning (i.e., learning features from raw data), and a hybrid of both to decide a final feature set. We also perform sensitivity analyses to decide different aspects of features (e.g., context window size, dimensionality, etc.).

1.3.3 Model Setup

Model setup refers to the process of selecting a learning model and setting its hyper–parameters. Model selection significantly depends on the complexity of a dataset. As an example, selecting a powerful model such as a neural network on linearly–separable data will result in over–fitting. For a given task, we use different sets of models including linear, ensemble, and deep models to decide a final model. We also exploit random search strategy [8] for hyper–parameter selection with 10–fold cross–validation for linear and ensemble models and a development (validation) set for deep models.

1.3.4 Knowledge Representation

Knowledge representation refers to a formal representation scheme of information in a way that computer software can utilize it to perform some tasks. The representation should support insertion, update, and querying operations. Nevertheless, defining a proper knowledge representation scheme for an arbitrary task is challenging. We use implicit knowledge representation (i.e., an internal representation of a trained model), distributional representation, predicate–argument structures, and semantic networks to represent learned aspects of common–sense knowledge.

1.4 Contributions

1.4.1 Spatial Role Labeling

We propose a joint sequence–to–sequence classification scheme based on deep Convolutional Neural Networks (CNN) and a hybrid set of universal and local features to automatically identify and assign roles to spatial signals within an input utterance. Considering the importance of spatial relations in describing 3D scenes, this contribution can significantly enhance the accuracy of semantic analysis performed by text–to–scene and text–to–animation systems. The results suggest that our approach significantly outperforms the state–of–the–art systems.

1.4.2 Learning Spatial Relations

The second contribution is towards grounding and inferring 3D spatial relations to position a model and identify spatial relations between two models in 3D space, respectively. We develop two deep feed–forward neural networks (DNN) based on a cognitive theory of spatial relations introduced in [9]. These DNNs use not only a set of view–centric geometric features (camera position, camera orientation, and object sizes) but also object knowledge (model annotation represented by continuous word vectors). Results suggest that training models using both geometric and semantic features enhance the performance. Most previous systems use a rule–based approach to address these tasks by defining a set of geometric templates [5], [10].

1.4.3 Learning Spatial Co–Occurrences

We also develop a recommender component to suggest a few objects and their relative spatial relations based on an object of interest to auto–complete a scene design. This component is a probabilistic graphical model trained on a database of 3D scenes. This capacity allows text–to–scene and text–to–animation systems to provide intelligent interactions between a user and the system.

1.4.4 Learning Physical Attributes of Objects

The fourth contribution is the automatic extraction of physical attributes of objects using supervised web content mining. We propose offline and online learning schemes to extract a set of corresponding values to learn the distributions of physical attributes of objects. Results suggest that the proposed framework can extract required physical attributes such as absolute size, mass, and speed of objects with high accuracy in soft real–time. Previous systems utilize unsupervised techniques such as meronym patterns [11], query templates [12], and hand–crafted patterns [13] to extract attributes and their corresponding values. These approaches are sensitive to threshold measures and setting them is not a trivial task. Also, they miss relevant information that is not predicted in engineered patterns. On the other hand, our approach learns the patterns and thresholds from data and hence achieves better results.

1.5 Organization

The rest of this document is organized into the following chapters:

- *Chapter 2* provides a concise background including terminology of computational linguistics and machine learning, engineered linguistic features, and the vector representation of words.
- *Chapter 3* presents a comprehensive overview of related systems and discusses various components used by them. It also provides in–depth quantitative evaluations on these systems.
- *Chapter 4* elaborates on our contribution regarding spatial role labeling and discusses the proposed CNN model, hybrid feature set, and utilized dataset.

- *Chapter 5* describes our two proposed neural models to infer and ground spatial relations, collected dataset, and proposed recommender system.
- *Chapter 6* elaborates on proposed offline–learning scheme to model physical attributes of various object categories using Gaussian Models.
- *Chapter 7* presents proposed CNN model to learn physical attributes of an arbitrary object in an online–learning scheme.
- *Chapter 8* concludes the thesis. It also outlines the directions for future work.

1.6 Published Papers

1) K. Hassani, W-S. Lee. "*Disambiguating Spatial Prepositions Using Deep Convolutional Networks*", Proceedings of the Thirty First AAAI Conference on Artificial Intelligence (AAAI-17), pp. 3209–3215, San Francisco, USA, 2017.
2) K. Hassani, W-S. Lee. "*Learning Physical Properties of Objects Using Gaussian Mixture Models*". Proceedings of the 30th Canadian Conference on Artificial Intelligence, pp. 179–190, Edmonton, Alberta, Canada, 2017.
3) K. Hassani, W-S. Lee. "Visualizing Natural Language Descriptions: A Survey". *ACM Computing Surveys*, 49(1), No. 59, 2016.
4) K. Hassani, W-S. Lee. "A Universal Architecture for Migrating Cognitive Agents: A Case Study on Automatic Animation Generation". In J. O. Turner, M. Nixon, U. Bernardet & S. DiPaola (Eds.), *Integrating Cognitive Architectures into Virtual Character Design*. IGI Global, Pennsylvania, USA. Chapter 9, 238–265, 2016.
5) K. Hassani, W-S. Lee. "*Adaptive Animation Generation Using Web Content Mining*". Proceedings of 2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS), pp. 1–8, Douai, France, 2015.
6) K. Hassani, W-S. Lee. "*On Designing Migrating Agents: From Autonomous Virtual Agents to Intelligent Robotic Systems*". Proceedings of ACM SIGGRAPH Asia 2014 Autonomous Virtual Humans and Social Robot for Telepresence, No. 7, Shenzhen, China, 2014.

7) K. Hassani, W-S. Lee. "*An Intelligent Architecture for Autonomous Virtual Agents Inspired by Onboard Autonomy*". Proceedings of the 7th IEEE International Conference on Intelligent Systems, pp. 391–402, Warsaw, Poland, 2014.

Chapter 2. Background

2.1 Terminology

Considering the interdisciplinary nature of this work and to provide the readers with a self–contained document, this section provides a concise terminology of natural language processing and machine learning.

2.1.1 Terminology of Natural Language Processing

Stop–Words: words with syntactic functionality that carry insignificant semantic information (e.g., "*the*" and "*is*") [14].

Bag–of–Words* (*BOW*) *Model: a text representation model that treats a given text as a set of word–frequency pairs and disregards syntax and word order [14].

***N–Grams*:** a contiguous sequence of *n* tokens from a given sequence of text [14].

***Skip–Grams*:** an *n*–gram model that allows tokens to be skipped to overcome data sparsity [15].

Word Embeddings: dense vector representation of words learned in an unsupervised manner [16]. They can capture fine–grained semantic and syntactic regularities using vector arithmetic and reflect similarities and dissimilarities between words [17].

Lemmatization: the process of grouping different inflected forms of a word so they can be analyzed as a single item (e.g., "*go*" is a lemma of: [*go*, *goes*, *going*, *went*, *gone*]) [14].

***Named–Entity Recognition* (*NER*)**: the process of locating and classifying elements in text into predefined categories (e.g., a person's name, organization, location, etc.) [14].

Part–Of–Speech* (*POS*) *Tagging: the process of labeling words in a given text by their grammatical category (e.g., noun, verb, etc.) [14].

Syntactic Parsing: the process of constructing a hierarchal syntactic tree of a given text to represents both word–level and phrase–level POS–tags (e.g., noun phrase, etc.) [14].

Semantic Parsing: the process of mapping a given text into a formal knowledge representation that can be processed by software [14].

***Semantic Role Labeling* (*SRL*)**: the process of identifying predicate–argument structures (e.g., verb–noun phrase) and assigning arguments with roles (e.g., agent, instrument, etc.) [18].

WordNet: an English lexical database that arranges words in an ontological representation based on some semantic relations (e.g., synonymy, hypernymy, etc.) [19].

FrameNet: an English lexical database containing manually annotated sentences for semantic role labeling [20].

ConceptNet: a semantic network of common–sense knowledge [21].

2.1.2 Terminology of Machine Learning

***Multi–Layer Perceptron* (*MLP*)**: a fully–connected feed–forward neural network that can estimate a mapping function between an arbitrary set of input–output pairs [22].

***Convolutional Neural Network* (*CNN*)**: a model of feed–forward neural networks inspired by visual cortex in which neurons arranged in 3D respond to restricted regions of space known as receptive fields [22].

***Recurrent Neural Network* (*RNN*)**: a model of neural networks with feedback connections exhibiting dynamic temporal behavior. Unlike feed–forward neural networks, RNNs can perform temporal processing and learn sequences [22].

***Long short–term memory* (*LSTM*)**: a model of RNN augmented with a *forget gate* that can learn tasks that require memories of important events with very long–time lags of unknown size between them [23].

Random Forest Classifier: an ensemble learning model that constructs a multitude of decision trees to mitigate a decision tree's habit of overfitting [24].

Bagging Classifier: a bootstrap ensemble method that trains classifiers on a random redistribution of a training set and then uses voting to select an output [25].

Ada–Boost Classifier: an ensemble model that combines outputs of *weak learners* into a weighted sum to represent a final output [26].

***Conditional Random Field* (*CRF*):** a discriminative undirected probabilistic graphical model used for structured predictions [27].

***Support Vector Machines* (*SVM*):** a supervised classifier that learns a set of hyper–planes in a high–dimensional space [28].

2.2 Engineered Linguistic Features

In most tasks, common–sense knowledge is represented purely or partially in natural language. Hence, we require linguistic analyses to extract a proper set of features to train a corresponding model. We use a set of morphological, syntactic, and semantic analyses to engineer linguistic features on a given corpus. Morphological and syntactic analyses are carried out using Stanford CoreNLP toolkit [29] to segment a data sample into a set of sentences, and to tokenize and lemmatize each sentence, accordingly. A POS–tagger [30] and a syntactic parser [31] are also utilized to extract constituents from a parse tree, and assign them with a set of syntactic tags (i.e., accuracy of 97.24% on Penn Treebank [32]). POS–tags follow the format of Penn Treebank tags [33] (i.e., illustrated in Appendix A). Using a hierarchy of a parse tree, we extract features related to phrase–level POS–tags of ancestors and siblings of a term.

The Stanford deterministic co–reference resolution system [34] is also used to resolve a reference to preceding constituents (i.e., anaphora resolution). As an example, for a sample such as "*Create a big wooden desk behind the piano and show a laptop on it. Also, hang a poster of Albert Eisenstein above it.*", extracted POS–tags, resolved co–references, and constructed parse tree (second sentence) are shown in Figures 2.1–2.3, respectively.

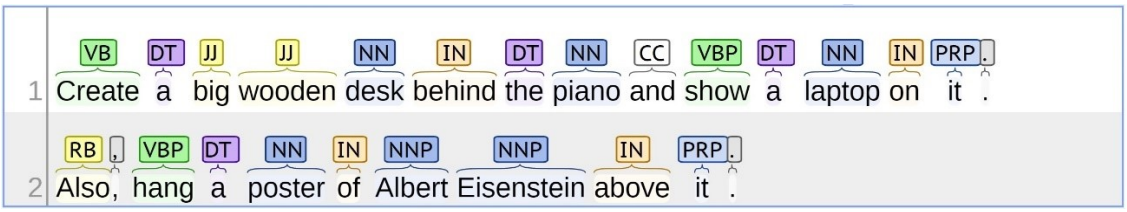

Figure 2.1. An example of POS–tagging using CoreNLP. POS–tags follow the format of Penn Treebank (e.g., VB: verb, DT: determiner, JJ: adjective, etc.)

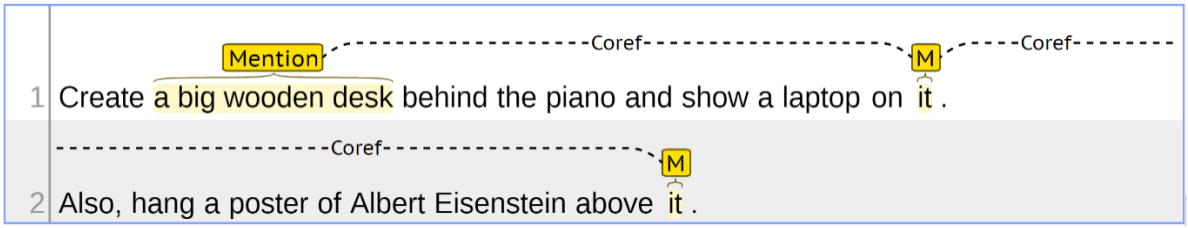

Figure 2.2. An example of co–reference resolution using CoreNLP. It is automatically detected that "*it*" refers to "*a big wooden desk*"

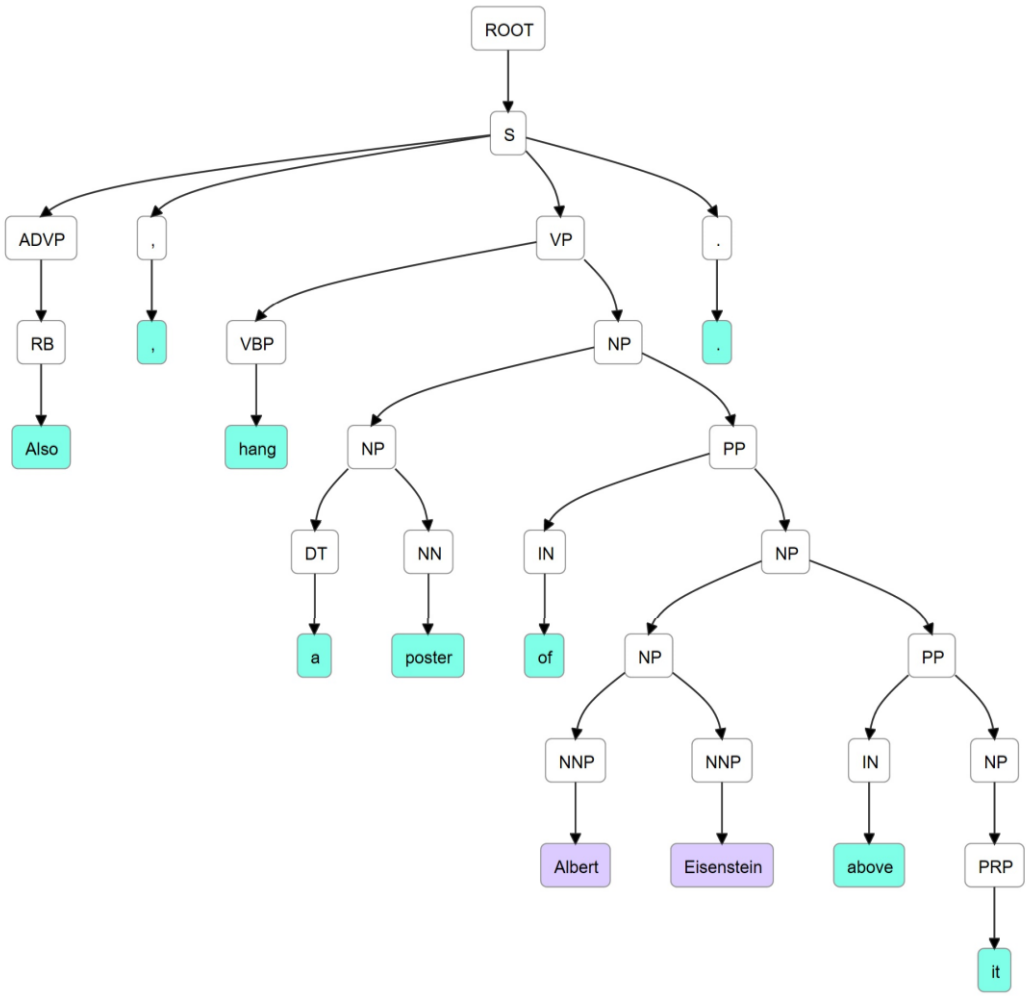

Figure 2.3. A sample parse tree generated by CoreNLP. In addition to word–level tags (e.g., DT, IN, NNP), a parse tree provides phrase–level tags (e.g., NP: noun phrase, PP: preposition phrase) and clause–level tags (e.g., S: simple declarative clause)

We also use Stanford Named–Entity Recognizer (NER) [35] to recognize persons, organizations, and locations (compound proper nouns) to increase the accuracy of *n*–gram models. An example of NER is shown in Figure 2.4.

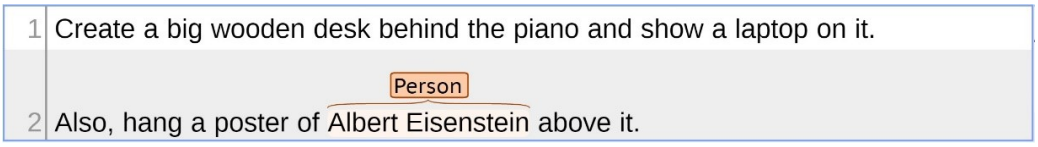

Figure 2.4. An example of named–entity recognition using CoreNLP. "*Albert Einstein*" is recognized as a person

Moreover, the Stanford dependency parser [36] is used to augment features. This parser constructs a shallow predicate–argument semantic representation by extracting relations between content words. The representation consists of a relation (predicate), and a governor and a dependent word (i.e., arguments). The parser can identify about 40 pre–determined relations (Appendix B) such as object, subject, modifiers, agent, etc. For instance, a dependency structure of a two–sentence sample: "*Create a big wooden desk behind the piano and show a laptop on it. A red book and a coffee mug are also beside the laptop.*" is shown in Figure 2.5.

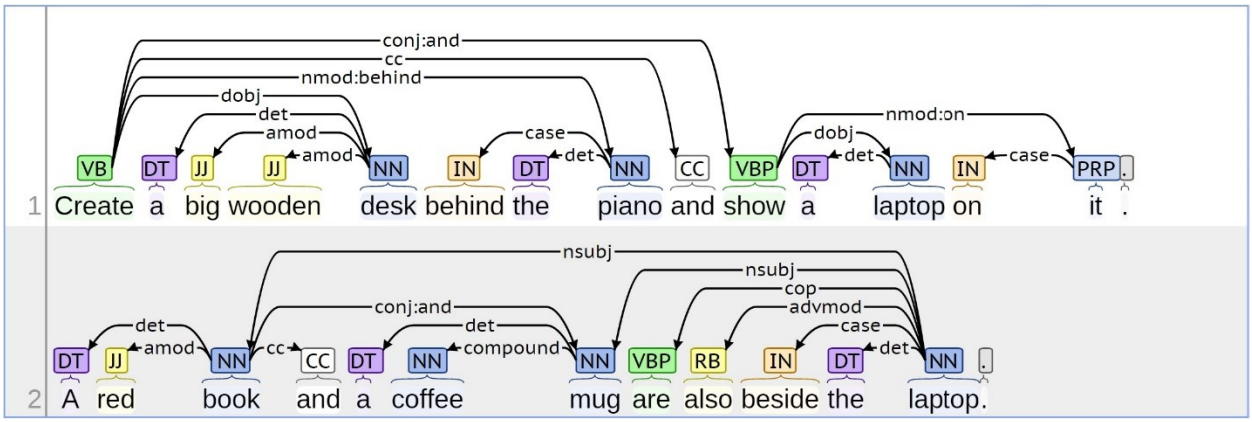

Figure 2.5. An example of dependency parsing using CoreNLP. A dependency structure consists of predicate–argument relations between two words. As an example, *dobj* (*show, laptop*) shows that "*laptop*" is a direct object of "*show*".

Finally, we utilize WordNet [19] hypernym hierarchy to detect whether a given word refers to a physical object or an abstract entity (i.e., a type of high–level binary word–sense disambiguation). We also use the number of senses of a word to compute a polysemy score. As an example, hypernym hierarchy of "*Guitar*" is as follows (it has only one sense in WordNet).

guitar -- (a stringed instrument usually having six strings; played by strumming or plucking) => stringed instrument -- (a musical instrument in which taut strings provide the source of sound) => musical instrument, instrument -- (any of various devices or contrivances that can be used to produce musical tones or sounds) => device -- (an instrumentality invented for a particular purpose; "the device is small enough to wear on your wrist"; "a device intended to conserve water") => instrumentality, instrumentation -- (an artifact (or system of artifacts) that is instrumental in accomplishing some end) => artifact, artefact -- (a man-made object

taken as a whole) => whole, unit -- (an assemblage of parts that is regarded as a single entity; "how big is that part compared to the whole?"; "the team is a unit") => object, physical object -- (a tangible and visible entity; an entity that can cast a shadow; "it was full of rackets, balls and other objects") => ***physical entity*** *-- (an entity that has physical existence) => entity -- (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))*

2.3 Word Embeddings

In addition to engineered linguistic features, we exploit linguistic features extracted using representation learning models. Word embeddings are dense vector representation of words learned in an unsupervised manner [16]. They can capture fine–grained semantic and syntactic regularities using vector arithmetic and reflect similarities and dissimilarities between words [17]. Like vector space models such as latent semantic analysis (LSA) [37], word embeddings are based on the distributional hypothesis (i.e., the meaning of a word can be determined by looking at its context) [38]. However, instead of global matrix factorization methods such as singular value decomposition (SVD), word embeddings are learned based on neural language models in which a word vector is a network's internal representation of a word. Because word embeddings are learned using shallow networks, learning them is much faster than matrix factorization methods [39], [40].

Several models such as Continuous Bag–of–Words (CBOW) and Skip–Gram with Negative–Sampling (SGNS) (also known as Word2Vec model) [41], [42], vector log–bilinear models (vLBL and ivLBL) [43], explicit word embeddings based on Positive Pointwise Mutual Information (PPMI) metric [44], and Global Vectors for word representation (GloVe) [17] are introduced in the literature. It is shown that if these models are trained on large corpora, the resulted vectors are universal word features that can be applied to various tasks [45]. In this study, we use pre–trained Word2Vec and GloVe word vectors.

2.3.1 Word2Vec Model

The Word2Vec embeddings are trained using skip–gram with negative–sampling (SGNS) model in a local context [41], [42]. This model is a single layer neural network that maximizes a sigmoid function of an inner product of a word and context vectors while minimizing negative

samples. These samples are generated by sampling a few words that do not appear in a context window. The loss function is defined as follows.

$$J = \frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{k=-|c|}^{|c|}\log p(w_{k+t} \mid w_t) \tag{2.1}$$

In this loss function, |V| denotes the vocabulary size, $|c|$ denotes the context window size, w_t is a word of interest, w_{k+t} is a set of $2\times|c|$ context words surrounding w_t and $p(w_{k+t}|w_t)$ is a skip–gram defined using a soft–max function. It is shown that SGNS implicitly factorizes a word–context matrix whose elements are Point–wise Mutual Information (PMI) of word–context pairs shifted by a constant [39]. Word2Vec pre–trained vectors are trained on part of the Google News dataset with about 100 billion tokens. The model contains 300–dimensional vectors for 3 million words and phrases and is publicly available[1].

2.3.2 GloVe Model

The GloVe model explicitly utilizes a word–context co–occurrence matrix. It leverages a combination of statistical information of global co–occurrence matrix with a weighted least squares regression model as follows [17].

$$J = \frac{1}{2}\sum_{i=1}^{|V|}\sum_{j=1|}^{|V|} f(C_{ij})(w_i^T w_j - \log C_{ij})^2 \tag{2.2}$$

|V| denotes the vocabulary size, C_{ij} denotes the number of co–occurrences of w_i and w_j, and $f(x)$ is a weighting function to smoothen rare and frequent occurrences defined as follows.

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & if\ x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

GloVe embeddings are trained on documents collected by Common Crawl with 840 billion tokens. The model contains 300–dimensional vectors for 2.2 million words which are publicly available[2].

[1] https://code.google.com/archive/p/word2vec/
[2] http://nlp.stanford.edu/projects/glove/

Chapter 3. Literature Review

We overview three general categories of systems developed to convert scripts in natural language into a visual content. These categories include text–to–picture, text–to–scene, and text–to–animation conversion systems. We mostly focus on how these systems address spatial relations in input utterances and how they acquire and apply common–sense knowledge. We also investigate how these systems address required characteristics such as interactivity and adaptivity. Finally, we compare and score them based on some diversity criteria.

3.1 Text–to–Picture

Text–to–picture approach treats the problem of mapping natural language descriptions to a visual representation as a data–driven image retrieval and ranking problem, and addresses it using foundations of web–based image search engines [46]–[49]. It utilizes text–mining techniques such as Bag–Of–Words (BOW) model to extract a set of representative keywords and uses them to retrieve and rank a set of images from an annotated image dataset using some semantic similarity measures. Keyword extraction can be augmented by filtering out those keywords that do not have a physical realization (i.e., not picturable keywords).

It assumes that an annotated image dataset is available. In the case of automatic annotation, it is a common practice to collect a repository of multi–modal information containing both images and text and then to use co–occurring text around an image to annotate it. In web–based image retrieval systems, this process is carried out by exploiting surrounding text of images and text appearing within HTML tags. Extracted text is then tokenized and a subset of terms is selected and scored to determine a set of weighted annotations of an image [50]–[54]. Extracted keywords are then matched against image annotations and a subset of images are retrieved and ranked for a given keyword based on some predefined similarity measures. Finally, for each keyword, retrieved images with highest ranks are illustrated in the same order that their corresponding concepts appear in the text.

This approach inherits the solid theoretical foundations of search engines. Because of exploiting statistical information retrieval rather than semantic analysis, it is computationally efficient [55]. However, it does not result in expected visualizations due to three reasons: (1) it cannot capture the semantic information in input descriptions; (2) it is restricted to a set of available images, and (3) it cannot interpolate the in–between visual information. Furthermore, because of these limitations, it cannot address spatial relations or other common–sense aspects, properly. This approach is not the focus of the thesis and hence only two systems are discussed.

3.1.1 Story Picturing Engine

Story picturing engine [46], [48] addresses mapping between a given textual story and a set of representative pictures by focusing on "*quantifying image importance in a pool of images*." [46]. It receives stories such as "*Vermont is mostly a rural state. The countryside has the cozy feeling of a place which…*" [46] and ranks a set of related images accordingly as output (i.e., each candidate image describes the whole story). It is a pipeline of three processes as follows. First, a set of descriptor keywords is extracted from an input story using dictionary–based stop–word elimination, and a combination of BOW (i.e., words with lower polysemy) and named–entity models. Second, a set of images containing at least one keyword and one named–entity is retrieved from a local image database. Finally, similarities between pairs of images based on visual and lexical features are estimated using Integrated Region Matching (IRM) distance [56], and then images are ranked and retrieved using a mutual reinforcement method. Despite its good accuracy and performance, it only retrieves one picture for a given story and ignores important aspects such as temporal and spatial relations.

3.1.2 Text–to–Picture Synthesis System

This system associates an extracted keyword with a relevant image and presents a story as a sequence of related pictures [47]. It approaches this task by optimizing the likelihood of extracted keywords, images, and placements given an input story. First, a set of nouns and adjectives are extracted using a POS–tagger and stop–words are eliminated. These words are fed to a logistic regression model to predict their probability of picturability based on Google web and image hit counts. The Text–Rank algorithm [57] is then applied to select a set of final

keywords. Images selection is carried out by matching the extracted keywords against available image annotations. Ultimately, the retrieved pictures are positioned based on three constraints: minimum overlap, the centrality of important pictures, and proportional closeness of pictures with respect to their associated keywords. An output of this system is illustrated in Figure 3.1.

Figure 3.1. A sample visual story generated by Text–to–Picture Synthesis System for: "*First the farmer gives hay to the goat. Then the farmer gets milk from the cow.*" (©AAAI 2007, reprint from [47])

3.2 Text–to–Scene

Text–to–scene approach directly renders an elaborated and unified static scene. It supports background, layout, lighting, objects, poses, relative sizes, spatial relations, and other features that cannot be addressed by a text–to–picture system [5]. In a text–to–scene conversion system, words with specific POS–tags carry more visual information. Nouns are associated with objects, adjectives are associated with object attributes, prepositions are mostly associated with spatial relations, and verbs usually determine actions and poses of articulated models.

This approach can generate elaborated and unified 3D visual content from a script within a single static scene which is a more coherent realization in comparison with text–to–picture approach. Nevertheless, it faces challenges such as identifying, disambiguating, and grounding spatial relations and extracting and modeling physical attributes of objects. Also, because a generated scene is static, it only visualizes a single episode and cannot address dynamics and temporal relations. This section presents an overview of seven text–to–scene conversion systems.

3.2.1 NALIG

Natural Language Driven Image Generation (NALIG) was one of the early projects on generating static 2D scenes from natural language descriptions [58], [59]. NALIG uses a very restricted form of input language (i.e., a simple regular expression) to investigate associations between spatial information and prepositions in Italian phrases. Allowed form of input phrases is a simple locative expression [60] as follows.

$$[Subject][\mathrm{P}reposition][Object] \tag{3.1}$$

NALIG can understand an input such as "*the book is on the table.*" It can also handle ambiguities within a phrase and infer simple implicit spatial arrangements using taxonomical rules such as "*Object X supports object Y"*. These rules are defined based on state conditions, containment constraints, structural constraints, and supporting rules. For example, given an input such as "*a branch on the roof*", it can infer that "*a tree near the house has a branch on the roof.*" In addition to spatial arrangements, NALIG utilizes statics (i.e., a branch of mechanics) to infer how an object can support another object based on a physical equilibrium. All in all, NALIG is a very restricted system that does not support user interactions, flexible inputs, or 3D spatial relations.

3.2.2 PUT

PUT [10] inspired by theories of cognitive linguistics is a rule–based spatial–manipulation system that generates static scenes through direct manipulation of spatial arrangements between rigid objects using a restricted subset of natural language. Using its restricted grammar, it can position 3D objects on top of each other or hang a 2D object on a 3D one. It can also disambiguate simple spatial relations. The syntax of its allowed language is in form of a locative expression as follows.

$$[V][TR][P\ LM]^{+} \tag{3.2}$$

V denotes a placement verb (i.e., only two verbs are supported: *put* and *hang*), TR denotes a trajector (located object), LM denotes a landmark (reference object), and P denotes a spatial preposition indicating a spatial relation between a trajector and a landmark. PUT contains

a set of 2D objects (e.g., *walls* and *rugs*) and a set of 3D objects (e.g., *tables* and *lamps*) that are manually scaled and included in a virtual world. Thus, a user is limited to a set of pre–existing objects and cannot insert new objects to the world. It also supports ten different groups of spatial relations (e.g., *above/below*, *left/right*, and *on*). The Kleene plus operator in (3.2) lets it handle compound spatial relations with a set of reference objects. As an example, an input utterance such as "*Put the box on the floor in front of the picture under the lamp*" is decomposed to:

$[put]_{V}$ $[the\ box]_{TR}$ $[on\ the\ floor]_{P–LM}$ $[in\ front\ of\ the\ picture]_{P–LM}$ $[under\ the\ lamp]_{P–LM}$.

3.2.3 WordsEye

WordsEye [5] is designed to generate 3D scenes containing environment, objects, characters, attributes, poses, kinematics, and spatial relations. It supports textual descriptions containing information about actions, spatial relations, and object attributes. WordsEye consists of two components: a linguistic analyzer and a scene depictor. Linguistic analyzer utilizes a statistical parser to construct a dependency structure which is used to construct a semantic representation in which objects, actions, and spatial relations are represented as semantic frames [61]. Nouns are associated with 3D models, spatial relations are captured using a set of predefined spatial patterns, and verbs are associated with a set of parameterized procedures.

At the core of WordsEye is Scenario–Based Lexical Knowledge Resource (SBLR) which is a knowledgebase tailored to represent lexical and common–sense knowledge [62]. Knowledge is represented as VigNets [63], [64] (i.e., an extension of FrameNet) consisting of a set of intermediate frames called Vignettes that bridge the semantic gap between semantic frames of FrameNet and low–level graphical frames. VigNet also contains implicit knowledge of a restricted set of environments (e.g., *kitchen*). This knowledge is a remedy for missing common–sense facts from a script in natural language. VigNet is populated by Vignettes extracted from a corpus collected from Amazon Mechanical Turk (AMT) [65] crowd–sourcing platform and processed using simple text processing techniques [63], [66]–[68].

Depiction module converts a set of semantic frames into a set of low–level graphical specifications. It uses a set of depiction rules to convert objects, actions, relations, and attributes from the extracted semantic representation to their realizable visual counterparts. This module also employs a set of transduction rules to solve implicit and conflicting constraints while positioning an object within a scene in an incremental manner.

WordsEye relies on its massive offline rule–base and data repositories. Its semantic database consists of 15,000 nouns and 2,300 verbs, and its visual database consists of 2,200 3D models and 10,000 images. 3D models are manually annotated with geometric information (e.g., shape, type, flexibility, embeddability, etc). As an instance, an object with a long thin vertical base is annotated as the *stem*. It also contains a large set of rules including spatial, depiction, and transduction rules. For example, it contains three rules for *kicking* action whose firing strengths are evaluated based on an object to be kicked. WordsEye has been utilized by a few thousand online users to create 15,000 static scenes. Although it has achieved a good degree of success, its allowed input language is stilted [61], [68]. WordsEye does not provide an interface to insert new knowledge and hence one requires coding them into it. A sample scene generated by WordsEye is illustrated in Figure 3.2.

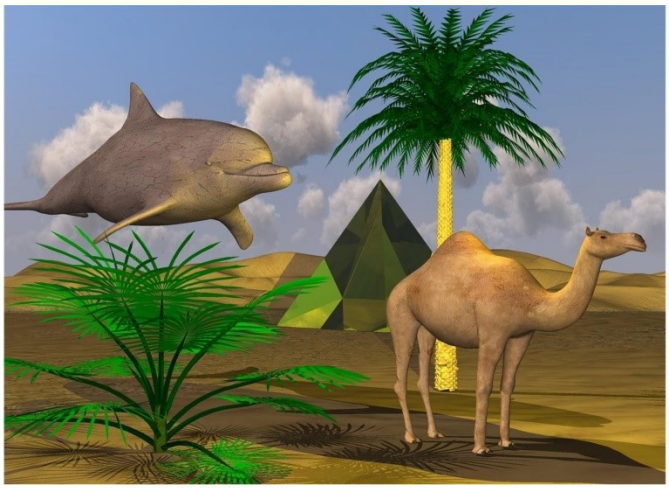

Figure 3.2. A sample output scene generated by WordsEys for the following description: "*The camel is in the desert. The small palm tree is 5 feet behind the camel. The huge dirt dolphin is behind the camel. It is left of the camel. It is above the camel. It is facing southeast. A 20-foot-tall transparent yellow pyramid is 50 feet behind the tree. The camel is facing southeast. A green palm tree is left of the camel.*" (Image courtesy of Wordseye Inc.)

3.2.4 AVDT

Automatic Visualization of Descriptive Texts (AVDT) [69] focuses on spatial relations to generate natural appearing 3D static scenes. It consists of two layers: automatic scene graph generation layer and object arranging layer. Former layer extracts information from a text and

generating a scene graph. It utilizes GATE [70] —an open–source text processing tool— to perform lemmatizing, POS–tagging, and dependency parsing. It assigns a meta–data to spatial prepositions and nouns and ignores the rest. A meta–data of a word contains its role (i.e., preposition, dependent, or supporter), position in the text, quantity, and corresponding 3D model (for nouns). Spatial prepositions are grouped with respect to their semantic similarities (e.g., *under*, *below*, and *beneath* are categorized as *under*). A directed graph is then constructed in which a node represents an object and an arc represents a preposition.

Object arranging layer uses this graph to render a scene. It assigns an AABB to an object and applies a few distance and rotation heuristics to standardize the scales and orientations of dependent and supporting objects. Rotation heuristic ensures a dependent faces its supporter and applies a little randomness to achieve an untidy appearance. AVDT focuses on the naturalness of a layout using hand–crafted heuristics and spatial analysis and results in more natural–looking scenes than WordsEye. AVDT can deal with linguistic cycles and allows more natural spatial language. As an example, AVDT outperforms WordsEye in visualizing a sentence such as "*On the table is a vase*" [69]. A sample scene generated by AVDT is illustrated in Figure 3.3.

Figure 3.3. A sample scene generated by AVDT for the following description: "*In front of a cottage is a tree. On the left side of the cottage are 2 trees and 3 trees are growing on the right side of the cottage. Behind the cottage is another tree. A bench is standing on the left side of the first tree and in front of the bench is a man. In front of the first tree is a fence. An old–timer waits in front of the fence. On the left side of the car is a lantern and another lantern is on the right side of the old–timer.*" (© Eurographics Association 2011; reprinted from [69])

3.2.5 Indoor Scene Generator

A system developed at the Stanford University [71]–[73] models implicit relations, support hierarchies, and explicit spatial relations using conditional probabilities. It constructs a scene template from an input text using CoreNLP toolkit [29]. This template is a graph with objects as nodes and spatial relations as arcs. An object is identified by detecting a noun that is a hypernym of the *physical entity* in WordNet. Adjectives within noun phrases are extracted to identify object attributes and spatial relations are extracted using a set of predefined patterns.

Descriptions in a natural language usually neglect a set of obvious facts about spatial arrangements as it is assumed that a target audience possesses the required knowledge to infer those facts. Nevertheless, like other aspects of the common–sense knowledge, inferring those facts is a challenging task for a computer. To address this issue, this system uses conditional probabilities to model the object occurrences and hierarchy priors and exploits Bayes' rule to infer the implicit spatial arrangements. Probabilities are extracted from an indoor scene dataset introduced in [74]. The inferred knowledge is then inserted into a scene template graph. Using this approach, it infers that an input text such as "*put the cake on the table*" is referring to "*put the cake on a plate and put the plate on the table*".

A geometric graph is then constructed based on a scene template which contains a set of 3D models corresponding to objects within a scene template and their associated spatial arrangements. 3D models are automatically scaled using an approach introduced in [75]. This graph is used directly to render a static 3D scene. This system also modifies its probabilistic model of support hierarchy by observing how users design a scene. For example, if a user asks the system to "*put a cup on the table"*, it increases the co–occurrence probability of a *cup* and a *table* and the probability of *"a table supporting a cup"*. This system surpasses knowledgebase systems in terms of knowledge inference and adaptive behavior. However, in terms of support for the spatial language, it inherits the problems of WordsEye and AVDT. A sample scene generated by this system is illustrated in Figure 3.4.

3.2.6 2D Visual Scene Generator

A promising data–driven system introduced in [76] exploits a set of semantic and visual features to jointly train a fully connected Conditional Random Field (CRF) classifier [27].

Figure 3.4. A sample scene generated by the system developed at the Stanford University for the following input: "*There is a room with a chair and a computer.*" Note that it automatically infers the presence of a *desk* and the fact that the *computer* should be supported by the *desk*.

The trained model is utilized to generate the most probable 2D scene given an utterance describing a scene. Scene annotations are collected from AMT. Semantic features are extracted in the form of predicate tuples using semantic role analysis [77], and occurrences, attributes, and positions are extracted as visual features. Associations between extracted predicate tuples and visual features are computed based on their co–occurrences using highest mutual information. Nevertheless, it is not clear how it performs in 3D scenarios. Three generated scenes are illustrated in Figure 3.5.

Figure 3.5. Three scenes generated by the system developed at Microsoft Research: (a) "*Jenny is catching the ball. Mike is kicking the ball. The table is next to the tree.*" (b) "*Mike is sitting next to Jenny. The cat is sitting next to the tree. Jenny is throwing the ball.*" (c) "*Mike is scared of lightning. It is a stormy day. Jenny is standing on the slide*" (reprint from [77])

3.2.7 AttribIt

AttribIt [78] is a modeling system designed to help a user create visual content using subjective attributes such as a *dangerous airplane*. It provides a user with a set of 3D parts of a model and helps him/her with assembling those components to construct a plausible model. For this purpose, a dataset is collected from AMT where a worker is presented with a set of 3D parts of a model (e.g., *airplane wings*) and is asked to compare pairs of models using adjectives. Associations between 3D parts and collected attributes are then ranked using Support Vector Machines (SVM). Trained model along with a GUI is used to provide a user with corresponding parts of a model. AttribIt is limited in terms of model parts and does not provide the user with direct manipulation of them. A snapshot of its interface is shown in Figure 3.6.

Figure 3.6. A snapshot of AttribIt's interface. It allows a user to create visual content using subjective attributes (reprint from [78])

Most of the early text–to–scene systems are very restricted in terms of allowed inputs, models, spatial relations, and interactivity. On the other hand, the knowledge–based text–to–scene systems rely on their massive databases and pattern sets to provide more flexibility. However, they require designers to manually provide the required information. Finally, developed data–driven systems suffer from the limited domain that they are designed for. We address these issues by developing more adaptive and general components for those aspects of commonsense knowledge that are required by these systems (e.g., physical attributes, etc).

3.3 Text–to–Animation

This family of systems supports dynamics and realizes temporal relations. It extends the text–to–scene approach by: (1) parameterizing and grounding visual verbs to a set of visual actions, (2) inferring motion trajectories, and (3) expanding relations to spatiotemporal relations. Mostly, it augments the knowledge–based text–to–scene systems with an action generator and a planner. In this section, seventeen text–to–animation conversion systems are discussed.

3.3.1 SHRLDU

SHRLDU [79] was one of the pioneer systems in integrating AI into computer graphics. It consists of a simulated robotic manipulator equipped with an intelligent controller that operates within a virtual toy world. The world contains a few blocks with different shapes, sizes, and colors. The robot can perform three actions on these blocks including: (1) moving a block to a location, (2) grasping a block, and (3) ungrasping a block. It manipulates a world based on restricted commands in natural language. SHRLDU consists of four modules including a language analyzer, planner, dialogue manager, and graphical engine.

The language analyzer is a syntactic parser that uses a world model to disambiguate an input command. In other words, it validates syntactic analysis with semantic clues acquired from the world. For an ambiguous command such as "*Put the red pyramid on the block in the box*", it first recognizes "*the red pyramid*" as a possible noun phrase and then checks the world model to determine whether a unique *red pyramid* exists. This observation is used to decide whether "*on the block*" is part of the noun phrase. Planner plans a sequence of feasible actions to reach a goal state using a backward chaining algorithm. Given that "*the red block is on top of the blue block*" in a world model and a user asks to "*put the pyramid on the blue block*", the robot first grasps the red block, moves it to a random location, ungrasps it, grasps the pyramid, moves it to top of the blue block, and ungrasps it. An interesting feature of SHRLDU is its dialogue manager which lets it answer simple queries regarding a world configuration and history of actions. It also can request clarification in case of ambiguities in an input command. Despite its restricted grammar and world simplicity, SHRLDU has inspired many systems. A sample scene of a reconstructed SHRLDU program is depicted in Figure 3.7.

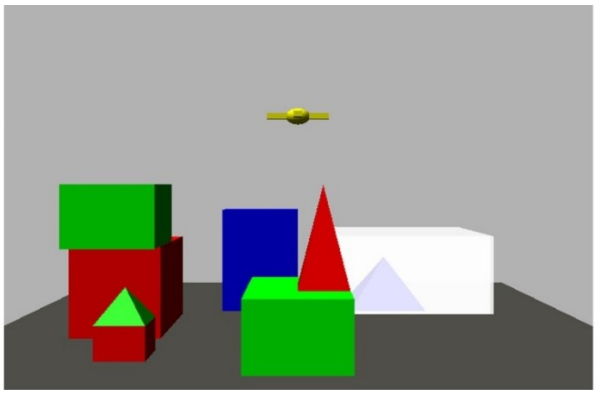

Figure 3.7. A snapshot from the SHRLDU's world

3.3.2 PAR

Parameterized Action Representation (PAR) is a framework for controlling virtual humans using natural language commands in a context–sensitive fashion [80]–[82]. It focuses on developing a knowledge representation scheme (shown in Eq. (3.3)) to reflect an input command on an agent's behavior [83]. This representation consists of following elements.

Applicability is a Boolean expression indicating the feasibility of an action for a given agent, and start and result indicate a set of states, time stamps, beginning, and termination of a given action. Participants denote an agent executing a PAR and a set of passive objects. Semantics include a set of Boolean pre–conditions and post–conditions of an action, path denotes a set of start and end points, direction, and distance of a motion. Purpose determines whether an action should satisfy a set of conditions or trigger another action, and termination determines a condition for terminating an action. The PAR structure also contains a set of pointers to other PARs including a parent, next, previous, and concurrent actions.

The execution architecture of PAR is a reactive framework that consists of five components including language converter, database manager, execution engine, agent process, and visualizer. Language converter parses an input command using XTAG parser [84] and uses a naive string matching to find a corresponding 3D model and an avatar. It also identifies a set of verbs and adjectives to construct a PAR representation. Execution engine synchronizes actions using its universal clock and passes a received PAR to an agent process. An agent within a world is assigned with an agent process that handles a queue of PARs (i.e., Pat–Net data structure) [85]. Visualizer renders a virtual world and its inhabitants based on a set of received commands from

the execution engine. The PAR architecture relies on shallow parsing rather than attempting to capture semantic information. It also does not support deliberative planning which is essential for generating plans for complicated goals. The lack of interactivity is another drawback of this architecture. A sample snapshot of an environment in which agents are controlled using PAR architecture is illustrated in Figure 3.8.

$$
PAR \Rightarrow \left[\begin{array}{ll}
\text{Applicability conditions} & : \text{Boolean} \\
\text{Start} & : \text{Time/State} \\
\text{Results} & : \text{Time/State} \\
\text{Participants} & : \left[\begin{array}{l}\text{Agent} \\ \text{Object set}\end{array}\right] \\
\text{Semantics} & : \left[\begin{array}{l}\text{Preconditions} : \text{Boolean} \\ \text{Postconditions} : \text{Boolean} \\ \text{Motion} : \left[\begin{array}{l}\text{Object} \\ \text{Caused} : \text{Boolean} \\ \text{Translational} : \text{Boolean} \\ \text{Rotational} : \text{Boolean}\end{array}\right] \\ \text{Force} : \left[\begin{array}{l}\text{Object} \\ \text{Contact point}\end{array}\right]\end{array}\right] \\
\text{Path} & : \left[\begin{array}{l}\text{Direction} \\ \text{Start} \\ \text{End} \\ \text{Distance} : \left[\begin{array}{l}\text{Units} \\ \text{Quantity}\end{array}\right]\end{array}\right] \\
\text{Purpose} & : \left[\begin{array}{l}\text{Achieve} : \text{Boolean} \\ \text{Generate} : \text{PAR} \\ \text{Enable} : \text{PAR}\end{array}\right] \\
\text{Termination} & : \text{Boolean} \\
\text{Duration} & : \left[\begin{array}{l}\text{Units} \\ \text{Quantity}\end{array}\right] \\
\text{Manner} & \\
\text{Subactions} & : \text{PAR constraint graph} \\
\text{Parent action} & : \text{PAR} \\
\text{Previous action} & : \text{PAR} \\
\text{Concurrent action} & : \text{PAR} \\
\text{Next action} & : \text{PAR}
\end{array}\right] \qquad (3.3)
$$

Figure 3.8. A sample snapshot of the Jack's Moose Lodge environment. In this world, agents are controlled using PAR structures (reprint from [80])

3.3.3 Carsim

Carsim [86]–[88] is a domain–specific system developed for generating simple animations of car accidents based on a set of Swedish accident reports collected from news articles, narratives from victims, and official transcriptions. It consists of two modules including information extraction and visualization modules. Information extraction module analyzes an input text and converts it to a triplet $<S,R,C>$ in which S denotes a set of scene objects (e.g., weather), R represents a set of road objects (e.g., cars), and C denotes a set of collisions. It utilizes the Granska POS–tagger [89] for tagging an input text and uses a small lexicon and a few regular expressions to extract a set of named–entities (e.g., street names). A light domain–specific ontology along with a classifier trained on a small set of example reports are employed to extract the events from textual descriptions of accidents.

The visualization module utilizes an animation planner and a graphical engine to render a planned animation. Animation planner exploits a greedy algorithm (i.e., backtracking is not supported) to plan an animation with respect to a set of constraints, initial positions and directions, and trajectories. Constraints are addressed using a small set of spatial and temporal rules. Initial direction and position are directly inferred from an input report and propagated to objects whose initial condition is not explicitly mentioned in the report. Trajectories are acquired using an Iterative Deepening A* (IDA*) algorithm. Carsim is a good example of a practical text–to–animation conversion system that mostly focuses on the practical aspects rather than

theoretical arguments. It has shown a fair degree of success in its limited domain. Yet, it lacks a solid mechanism to harvest the information from user interactions and feedbacks. It also does not contain a strong object repository or lexical resources.

3.3.4 ScriptViz

ScriptViz [90] is developed to replace manual storyboard drawing with automatic scene generation in a motion picture production process. It can analyze screenplays written in well–formed sentences (i.e., grammatically correct and not ambiguous) and animate a set of corresponding objects, agents, and actions. It consists of three interacting modules including language understanding, high–level planner, and scene generator modules. The language understanding module uses the APP parser [91] to derive a syntactical structure of an input text. Planner generates action plans based on information provided by language module as follows. First, an offline plan outline is extracted from a plan database with respect to objects and actions detected in an input script, and their states are collected from a scene and are used to decide the feasibility of an action. In the case of a feasible action, parameters of the offline plan are set and the result is represented using PAR structure [83]. Scene generator assigns a PAR to an agent, updates the states, and renders a scene in real–time. This system does not support lexical and common–sense resources.

3.3.5 CONFUCIS

CONFUCIS [7] is a multi–modal narrator system that generates an animation from a single input sentence containing an action verb. It can address temporal relations between actions performed by virtual humans and supports lip synchronization and facial expressions [92]. It consists of a knowledgebase, language processor, media allocator, animation engine, and synchronizer. The knowledgebase contains a lexicon and a visual database. Language processor uses Connexor's parser [93], WordNet [19], and a conceptual database [94] to parse an input sentence. The media allocator represents three modalities including animation, speech, and narration in XML format. The animation engine and synchronizer generate the animation and synchronize it with speech.

A main challenge of text–to–animation approach is defining actions that yield in a high–level realization. Assuming an input utterance such is "*John hits Paul with a bottle and John is in a distance of 2m from Paul and there is a bottle on a table that is in a distance of 1m from John*", a system should use a planner to plan a set of intermediate actions such as "*John walks towards the table*", "*picks up the bottle*", "*walks towards Paul*", and "*hits him with the bottle*". CONFUCIUS addresses this by hand–crafting actions which restrict it to a few predefined actions (less than 20 visual verbs). A snapshot of a sample animation is depicted in Figure 3.9.

Figure 3.9. A snapshot of a sample animation generated by CONFUCIS for the following sentence: "*John put a cup on the table*" (reprint from [92])

3.3.6 Scene Maker

SceneMaker [95], [96] is a collaborative and multi–modal system designed for pre–visualizing a given script to facilitate movie production. It is a successor of CONFUCIS and exploits its underlying language processing and animation generation tools. It expands CONFUCIS by adding common–sense knowledge of the genre and emotional expressions and allowing users to edit an animation via mobile devices. It consists of two layers including a user interface and a scene production layer. The user interface provides a user with a 3D animation and lets her/him edit it. Scene production layer is a pipeline of three modules including understanding, reasoning, and visualization modules. The understanding module performs text analysis, reasoning module uses WordNet–Affect [97] —an extension to WordNet [19]— and ConceptNet [21] to interpret a context, manage emotions, and plan actions. The visualization

module retrieves a set of 3D models and music from a database, generates speech, and sets the camera and lighting configuration.

3.3.7 Motion Generator

This system is designed to generate motion for virtual agents using a set of motion clips stored within a motion database [98], [99]. It exploits motion frames —an extension of case frames focusing on semantic valence [100]— as its knowledge representation scheme. A motion frame consists of an agent, a motion, an instrument, a target, a contact position, a direction, an initial posture, and a set of motion modifiers. It assumes that characters, objects, and motion frames are manually predefined by a user.

An input sentence is parsed using the CoreNLP tool [29] and then a small set of temporal rules and a verb dictionary are utilized to extract a set of query frames and temporal constraints. The motion database consists of a set of manually annotated atomic motions represented as motion frames. A query frame is searched within the motion database, and retrieved candidates are then ranked using a weighted similarity measure based on target, instrument, initial posture, and motion modifiers. Ultimately, a set of retrieved atomic motions is integrated to generate compound motions. This system relies on its offline motion database which makes it difficult to handle unseen motions. A sample sequence of a generated animation is shown in Figure 3.10.

Figure 3.10. A sample animation generated for the following script: "*Neo waves to Jack. At the same time, Jack takes the red bottle. Jack hits Neo with it*" (reprint from [99])

3.3.8 IVELL

Intelligent Virtual Environment for Language Learning (IVELL) [101], [102] is a domain–specific distributed virtual reality system that uses a few Embodied Conversational Agents (ECA) to improve the speaking skills of non–native users in English. It implements a few scenarios such as an *airport* and a *shopping mall* in which learners speak to an agent (e.g., immigration agent). An agent consists of a language interpreter, user evaluator, fuzzy knowledgebase, and language generator. Language interpreter parses an utterance using the OpenNLP tool. User evaluator uses a weighted model to score a user's proficiency. Knowledgebase is a light domain–specific fuzzy ontology that contains some knowledge about a predefined set of tasks. The language generator produces a set of answers with different difficulty levels. A sample interaction between a user's avatar and an agent is shown in Figure 3.11.

Figure 3.11. A sample interaction between an agent and a user's avatar in IVELL

3.3.9 Other Systems

The story Driven Animation System (SDAS) [103] supports a very limited set of simple articulated figures and primitive joint motions to create a simple animation based on a given unambiguous script. It consists of three components including story understanding, stage directing, and action generating components. The story understanding component performs Japanese syntactic analysis and uses an assumption–based reasoning to add very simple implicit assertions about the story. The stage direction module exploits a few simple heuristics to position

the actors and set the background based on extracted information and generated assertions. Action generating module uses a set of model descriptions and motion descriptions to produce some simple motions.

3DSV [104]–[107] creates an interactive interface for animating 3D stages of simple stories described in a restricted form. A stage includes objects, their attributes, and simple spatial relations. 3DSV utilizes an XML–based knowledgebase containing visual descriptions of a predefined set of objects, attributes, and spatial relations to parameterize a stage. Information extracted from an input story is integrated with available knowledge and then is converted to a VRML format.

Interactive e–Hon [108], [109] is a multi–modal storytelling system to facilitate interactions between parents and children by animating and explaining difficult concepts in a simpler form. This system uses a Japanese morphological analyzer and a lexicon to extract time, space, weather, objects, and actions from a story and then matches them against some predefined sets. Time, space, and weather are matched against a background set to provide an appropriate static background. Objects and actions are matched against an action table that is used to retrieve a corresponding recorded animation from a database.

A semi–automatic system introduced in [110], [111] creates animations from a fiction script assuming that a set of characters, objects, environment configuration, spatial relations, and character transitions are annotated in a well–formed structure in advance. It uses annotations of characters and objects to query a 3D model database and uses annotated relations to position the retrieved models.

A data–driven system developed at the University of Melbourne [112] trains a maximum entropy classifier [113] to ground high–level verbs into a set of low–level graphical tasks. It jointly trains a set of linguistic and stage features to predict a next graphical action. The linguistic features consist of a set of verb features, collocation features, and semantic role features whereas stage features are a set of binary spatial features extracted from a virtual stage. Snapshots from a sample animation generated by this system are shown in Figure 3.12.

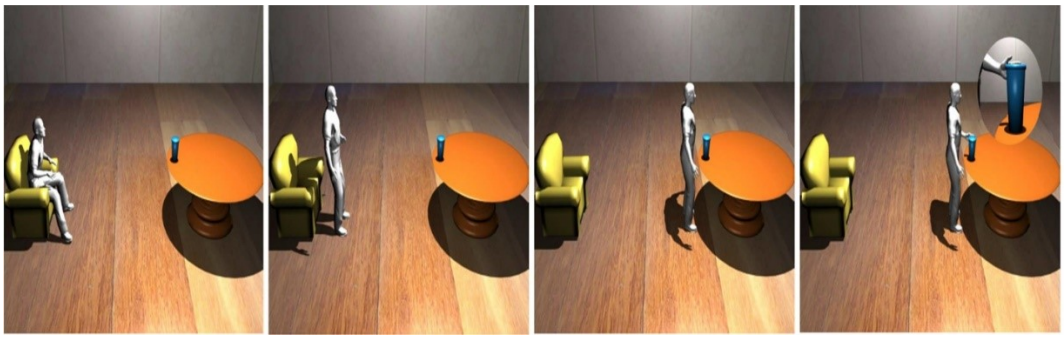

Figure 3.12. A sample animation generated by a system developed at the Melbourne University for "*The man grabs the mug on the table*" (©AAAI, 2008) (reprint from [112])

Web2Animation [114] is a multi–modal pedagogical system that uses the web content to create an online animation to teach a user how to cook. It collects a set of HTML documents related to a given recipe and then locates the relevant information by traversing a specific set of HTML tags and analyzing the content using the Phoenix parser [115]. The extracted instructions and ingredients are mapped into a few actions and objects, respectively. A domain–specific ontology is also utilized to match an action with its graphical representation. Finally, the generated animation is synchronized with a monolog explaining a recipe.

Vist3D [116], [117] is a domain–specific system for creating a 3D animation of historical naval battles from narratives. It uses a manually populated ship specification database and a temporal database. The temporal database is populated by a narrative analyzer that extracts time, date, and [Subject][Verb][Object] structures using a set of regular expressions. The retrieved information is then directly mapped to a VRML file.

A different approach that relies on multi–agent design is proposed in [118]. It models an agent using NLP4INGENIAS [119] (i.e., a multi–agent system based on INGENIAS framework [120]) and passes it to Alice [121] —a rapid prototyping environment for generating virtual environments— to render the agent within an arbitrary world.

Most of the text–to–animation systems follow a similar knowledge–based approach that text–to–scene systems follow. They utilize similar databases and set of patterns to address the input interpretation, 3D models, and physical attributes. They also use a set of limited and pre–determined actions. We enhance these systems by automatically predicting their dynamic attributes such as *mass* and *speed*.

3.4 Evaluation

The evolution of text–to–scene systems is investigated in terms of five measures: lexical, syntactic, action, spatial, and model diversities. Lexical and syntactic diversity measures are related to allowed utterances (spatial expressions) whereas other three measures determine the quality of generated visual content. These measures are defined based on the Likert scale and have five distinct values including: +2 (very high), +1 (high), 0 (medium), −1 (low), and −2 (very low). We compute these measures based on the diversity of a system compared to other systems. The evolution timeline for text–to–scene family is illustrated in Figure 3.13.

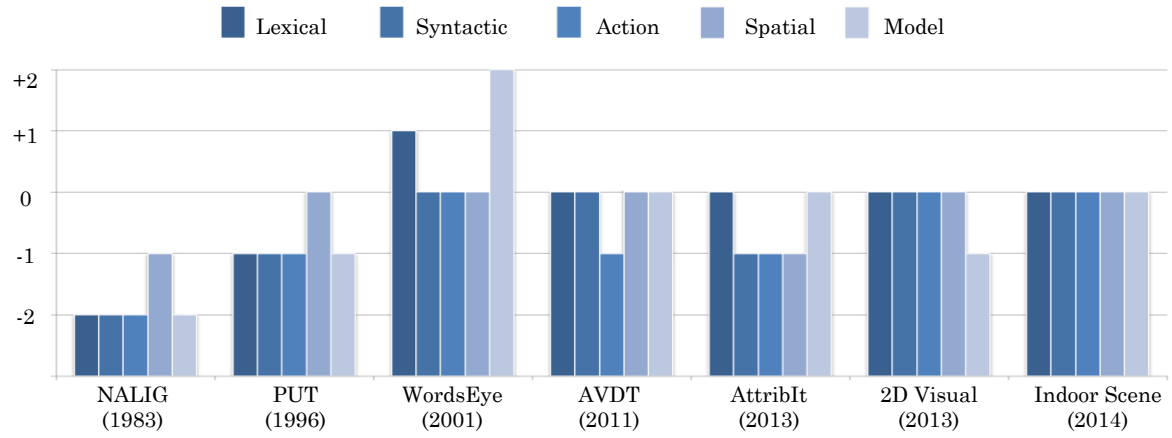

Figure 3.13. Evolution of text–to–scene conversion systems measured based on the lexical, syntactic, action, spatial, and model diversity

As shown, diversity of allowed utterances improves from NALIG to WordsEye and then stops. This trend reflects the current technical difficulties in natural language understanding techniques. We use a neural learning model to overcome this difficulty in spatial utterances. The evolution of action diversity follows a similar trend. Because of the vast variety of possible actions, it is not practical to craft them. On the other hand, learning actions and associating them with action verbs is a big challenge for current machine vision techniques. Furthermore, because most systems use a template–based approach, they do not achieve a good performance in terms of spatial relations. We address this by developing two neural models to infer and ground the spatial relations. In terms of model diversity, WordsEye achieves a good performance by relying on its huge 3D model database and a large number of hand–crafted model annotations.

An important observation is that current data–driven systems do not outperform knowledge–based systems. This is probably because data–driven systems have been only used for feasibility studies whereas a few knowledge–based systems such as WordsEye are commercialized.

Evolution of text–to–animation conversion systems is investigated with a similar approach by adding a temporal diversity measure. We similarly score the systems based on their diversity compared to other text–to–animation systems. This evolution is illustrated in Figure 3.14.

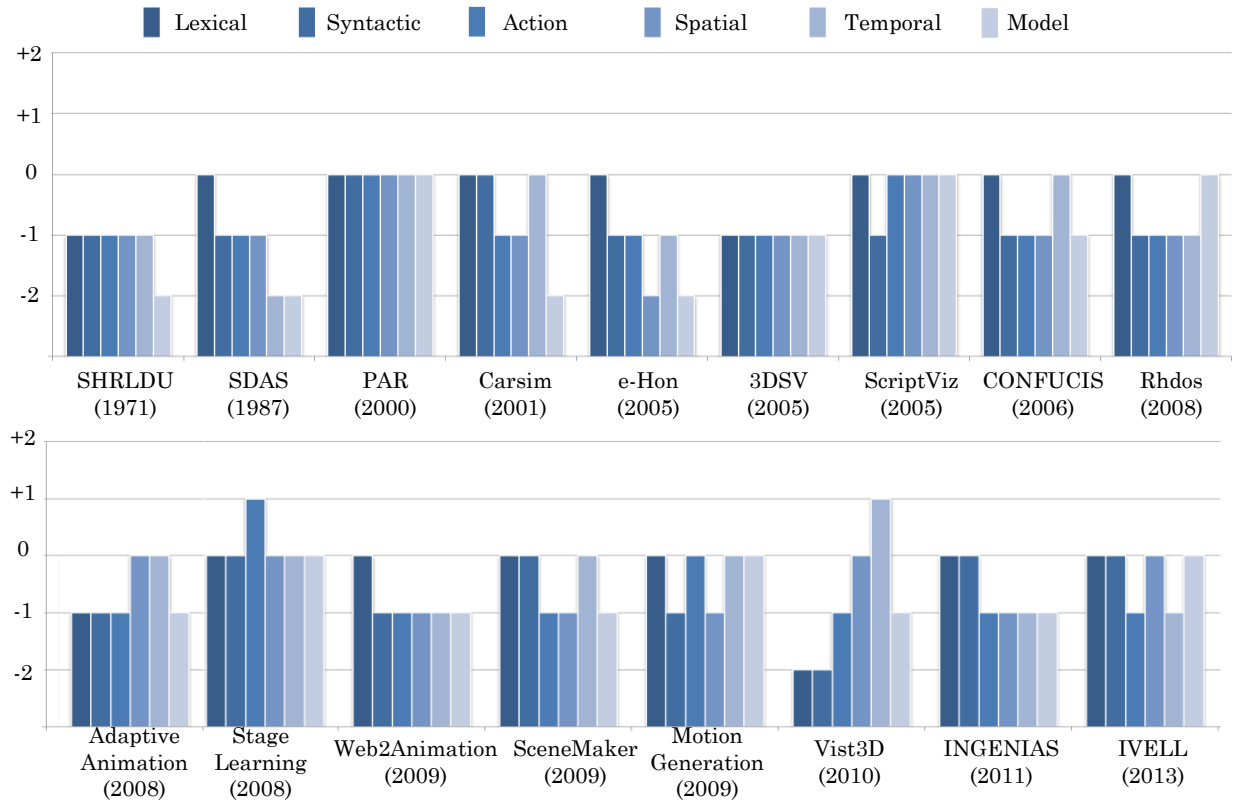

Figure 3.14. Evolution of text–to–animation conversion systems in terms of lexical, syntactic, action, spatial, temporal, and model diversity

Surprisingly, as shown in Figure 3.14 text–to–animation conversion systems have not improved much since SHRLDU. These systems can improve in terms of model diversity using similar approaches taken by systems such as WordsEye. Also, because temporal relations are more limited than spatial relations, this measure can be improved using temporal templates.

Nevertheless, these systems inherit the same challenges related to actions, spatial relations, and allowed utterances.

The discussed systems are summarized in Table 3.1. In this table, type indicates whether a system is a text–to–picture, text–to–scene, or a text–to–animation conversion system. Interactivity indicates whether it provides a user with means to manipulate a generated output, whereas adaptivity refers to its ability in extracting and inferring information that is not given in advance (data–driven systems are adaptive). The interface can be text, speech, or pointer (i.e., mouse interaction). Domain determines whether a system is developed for a general or specific purpose. Syntactic and semantic analyses and knowledgebase refer to utilized language understanding approach and knowledge resources. Finally, methodology indicates whether a system is data–driven, knowledge–based, rule–based, or multi–agent.

In terms of system behavior, 30.7% of them are interactive and 42.3% of them are adaptive. This reveals a fundamental flaw. Considering the current technical challenges in natural language understanding, a system should allow a user to interact with it and harvest relevance feedbacks to evolve in an incremental manner. It can disambiguate an input utterance in collaboration with a user as well. Also, considering the huge amount of required common–sense knowledge, it is not practical to gather it manually. Therefore, data–driven methods (e.g., web content mining, etc.) and active learning (i.e., user–in–the–loop learning) should be integrated into these systems.

In terms of an interface type, only 7.7% of systems utilize mouse interactions and only 3.8% of them utilize speech. A recent Human–Computer Interaction (HCI) comprehensive user study [122] suggests that natural language interfaces outperform graphical interfaces in terms of high–level control over virtual objects and animation design, whereas it is simpler to use graphical interfaces for spatial and motion control. It concludes that graphical interfaces increase the creativity in micro–level design while in macro–level design natural language interfaces are more efficient because of their higher versatility. It is also shown that natural language interfaces significantly reduce both learning time and design time. Thus, a good strategy is to develop a multimodal interface allowing a user to choose an interaction modality for his/her comfort. Furthermore, considering the current advances in speech recognition technology, it is simpler to use speech rather than typed text.

Table 3.1. Characteristics of the language–based visualization systems

System	Type	Interactive	Adaptive	Interface Type	Domain	Syntactic Analysis	Semantic Analysis
Story Picturing	TTP[a]	✗	✓	Typed text	General	BOW	Association analysis
TTP Synthesizer	TTP	✗	✓	Typed text	General	POS–tagging	Association analysis
PUT	TTS[b]	✗	✗	Typed text	General	Regular expression	Naive matching
WordsEye	TTS	✗	✗	Typed text	General	Statistical parsing	Dependency analysis
AVDT	TTS	✗	✗	Typed text	General	Statistical parsing	Dependency analysis
Indoor Scene	TTS	✓	✓	Text + Pointer	General	Statistical parsing	Naive matching
NALIG	TTS	✗	✗	Typed text	General	Regular expression	Naive matching
Attriblt	TTS	✓	✓	Typed text	General	POS–tagging	Association analysis
2D Visual Content	TTS	✗	✓	Typed text	General	POS–tagging	Semantic role analysis
Stage Learning	TTS	✗	✓	Typed text	General	Statistical parsing	Semantic role analysis
SHRLDU`	TTA[c]	✓	✗	Typed text	Specific	Dependency Parsing	Deep semantic parsing
CONFUCIS	TTA	✗	✗	Typed text	General	Dependency parsing	Lexical structure
SceneMaker	TTA	✓	✗	Typed text	General	Dependency parsing	ConceptNet
ScriptViz	TTA	✗	✗	Typed text	General	Statistical parsing	Naive matching
Motion Generation	TTA	✗	✗	Typed text	General	Statistical parsing	Naive matching
Carsim	TTA	✗	✗	Typed text	Specific	POS–tagging	Ontology
PAR	TTA	✗	✗	Typed text	General	Dependency parsing	Naive matching
IVELL	TTA	✓	✓	Speech + Pointer	Specific	Statistical parsing	Naive matching
SDAS	TTA	✗	✗	Typed text	General	Unknown	Unknown
Adaptive	TTA	✓	✓	Typed text	Specific	Regular expression	Naive matching
Interactive e–Hon	TTA	✗	✓	Typed text	General	Regular expression	Naive matching
Vist3D	TTA	✗	✗	Typed text	Specific	Regular expression	Naive matching
Web2Animation	TTA	✗	✓	Typed text	Specific	Statistical parsing	Ontology
Rhodes	TTA	✓	✓	Typed text	Specific	Regular expression	Naive matching
3DSV	TTA	✓	✗	Typed text	Specific	Regular expression	Naive matching
INGENIAS	TTA	✗	✗	Typed text	General	Statistical parsing	Active user involvement

[a] Text–to–Picture. [b] Text–to–Scene. [c] Text–to–Animation.

System	Knowledgebase + Lexicon	Methodology	References
Story Picturing Engine	WordNet	Data–driven	[46], [48]
Text–to–Picture Synthesizer	——	Data–driven	[47]
PUT	——	Rule–based	[10]
WordsEye	VigNet + WordNet	Knowledge–based	[5]
AVDT	——	Rule–based	[69]
Indoor Scene	WordNet + Commonsense	Data–driven	[71], [71], [72]
NALIG	——	Rule–based	[58], [59], [123]
AttribIt	——	Data–driven	[78]
2D Visual Content	——	Data–driven	[76]
Stage Learning	——	Data–driven	[112]
SHRLDU`	——	Rule–based	[79]
CONFUCIS	WordNet + Conceptual database	Knowledge–based	[7][92]
SceneMaker	WordNet–Affect + ConceptNet	Rule–based	[95], [95], [96]
ScriptViz	——	Rule–based	[90]
Motion Generation	——	Rule–based	[98], [99]
Carsim	WordNet + Ontology	Knowledge–based	[86]–[88]
PAR	——	Rule–based	[80], [82]
IVELL	Fuzzy ontology	Multi–agent	[101], [102]
SDAS	——	Rule–based	[103]
Adaptive Animation	WordNet	Data–driven	[124]
Interactive e–Hon	——	Rule–based	[108], [109]
Vist3D	——	Rule–based	[116], [117]
Web2Animation	Ontology	Rule–based	[114]
Rhodes	WordNet	Rule–based	[110], [111]
3DSV	XML–based knowledgebase	Rule–based	[105]–[107], [125]
INGENIAS–based System	——	Multi–agent	[118]

Among the systems, 69.2% of them are general–domain whereas 30.8% are designed for specific domains. 26.9% of them follow a data–driven approach, 7.7% follow a multi–agent paradigm, 11.5% are knowledge–based, and 53.8% are rule–based. Surprisingly, 61.1% of general–domain systems are rule–based. In terms of syntactic analyses, 26.9% of them exploit regular expressions, 15.4% of them utilize POS–tagging, and 34.6% of them employ syntactic parsing. Among general–domain systems, 61.1% of them use syntactic parsing, 16.7% of them exploit POS–tagging, and 16.7% of them use regular expressions. In terms of semantic analyses, 46.2% of them rely on naive keyword matching, 53.8% exploit some shallow semantic analyses (i.e., 33.3% of general–domain systems), 15.4% use some knowledgebase and ontology, and 7.7% exploit user–in–the–loop semantic analysis. Finally, in terms of knowledgebase (e.g., lexicon, ontology, etc.), 57.7% of them ignore these resources. This ratio is 72.2% for general–domain systems which naturally require common–sense knowledge resources. This fact highlights another fundamental problem: most systems ignore knowledge resources and hence cannot behave properly in unpredicted situations. The most frequently used lexical resource is WordNet [19] (i.e., 63.6% of systems).

Finally, most systems use pre–defined templates to handle spatial relations in both language space and visual space. Nevertheless, considering the complexity of spatial relations in both spaces, template–based approach fails in many cases. Also, most systems annotate a set of 3D models in advance. However, this manual work is not practical for larger 3D model databases. To address these issues, we exploit deep learning and probabilistic graphical models to address two important aspects of common–sense knowledge including spatial relations and physical object attributes.

Chapter 4. Spatial Role Labeling

The introduced visualization systems in chapter 3 use either a set of regular expressions or a set of syntactic rules to capture a limited set of spatial expressions. Hence, they cannot handle the spatial expressions in unrestricted utterances and as a result restrict the user's inputs. To address this limitation, we first investigate the task of preposition disambiguation task and show that deep models outperform other models on this task. We then investigate the semantic role labeling of spatial signals and show that deep models outperform other sequence–to–sequence models. We also investigate the effect of different features sets on these sets. To decide the final learning mode, we fix a deep architecture and then optimize its hidden units. We then compare its performance with other ensemble and linear models.

Evolution has shaped complex visual–spatial processing capabilities such as object recognition, object search, and navigation through space almost in all advanced species. It has also equipped humans with a discriminative ability to express and communicate spatial knowledge through language [126]. One of the ambitious goals of AI is to simulate this cognitive process to improve the applications of spatial knowledge. These applications include, but are not limited to, human–robot interactions, natural language interfaces, machine vision, text–to–scene conversion systems, geographical information systems (GIS), question answering systems, search engines, and spatial databases. This process entails various underlying cognitive–linguistic sub–processes such as detecting, representing, grounding, planning, and inferring spatial relations.

From a linguistic point of view, given an input utterance such as "*Put the laptop on the table to the right of the blue book*", the first step towards simulating this process is to decide whether a given utterance carries any spatial information. This step is carried out by detecting embedded spatial signals within an utterance (i.e., in this case "*on*" and "*to the right of*"). These signals are referred as pivot spatial signals (spatial indicators). In English, these signals can employ various syntactic categories such as verbs ("*leave"*), adverbs ("*back"*), adjectives ("*close"*), nouns ("*front"*), pronouns ("*here"*), and prepositions ("*on"*) to express complex

spatial relations between objects, motions through space relative to some reference point, and passive trajectories.

Spatial prepositions are strong discriminative signals in utterances indicating the existence of spatial information. They can be classified into locative and directional prepositions. Locative prepositions describe a position of a located object in relation to a set of reference objects whereas directional prepositions describe a change of position or direction of a located object. Locative prepositions are categorized to projective and topological prepositions. Projective prepositions such as "*above"*, "*in front of"*, and "*to the left of"* stipulate information regarding direction of a located object with respect to a reference object. Topological prepositions such as "*in"*, "*on"*, and "*near"* convey information about topological arrangements among objects. They can be further classified to simple topological prepositions such as "*in"* and "*on"* and proximity topological prepositions such as "*near"* and "*far from"* [9].

English has a limited number of prepositions (about 150) out of which some can carry spatial signals (about 80). Given this limited number of spatial prepositions, one can identify them using a lookup a table. However, prepositions are polysemous in nature and usually have many senses [127], [128]. For example, "*upon"* has a total of 25 senses out of which only 8 are spatial. Therefore, it is necessary to disambiguate senses of a candidate spatial preposition within a given context.

An extracted pivot signal can be assumed as a predicate and hence the next step is to identify its predicates (e.g., "*laptop*", "*table*", and "*blue book*") and annotate them with some spatial roles (e.g., *reference object* and *located objected*). Spatial prepositions appear in locative expressions conveying information about a spatial configuration of two or more objects in some space. A locative expression [60] consist of a preposition, its objects, and a subject that a prepositional phrase modifies. An object of a preposition is called landmark, anchor, reference object, or relatum, and its subject is referred as trajector, figure, located object or locatum. A locative expression describes a location, trajectory, orientation, direction, or disposition of a relatum in relation to a locatum, and a preposition describes the nature of this relationship [129]. Throughout this document, we will use ***pivot***, ***locatum***, and ***relatum*** to address the spatial predicates and arguments. Using this notation, the goal is to extract a set of predicate–arguments such as ***P:****on*→<***L**:laptop , **R**:table*> for a given utterance such as our example. As discussed in chapter 3, this can significantly enhance the allowed utterances.

Many spatial expressions in English and other languages are in the form syntactic frames. Three of such frames mentioned in [130] are as follows:

$$\mathbf{L_1} = [\text{NP}][\text{Existance Verb}][\text{Preposition}][\text{NP}] \tag{4.1}$$

$$\mathbf{L_2} = [\text{NP}][\text{Activity Verb}][\text{Preposition}][\text{NP}] \tag{4.2}$$

$$\mathbf{L_3} = [\text{NP}][\text{Trajectory Verb}][\text{NP}] \tag{4.3}$$

We can further augment these syntactic frames to address more general spatial expressions as follows.

$$\mathbf{L_4} = \left\{[\text{NP}][\text{Existance Verb}]^K[\text{Preposition}][\text{NP}] \quad ,0 \le K \le 1\right\} \tag{4.4}$$

$$\mathbf{L_5} = \left\{[\text{NP}][\text{Motion Verb}]^K[\text{Preposition}][\text{NP}] \quad ,0 \le K \le 1\right\} \tag{4.5}$$

$$\mathbf{L_6} = \left\{[\text{NP}][\text{Trajectory Verb}][\text{NP}]\right\} \tag{4.6}$$

Some examples for these syntactic frames are as follows.

(L_4 and K=1): [*The big spider*] $_{\text{NP}}$ [*is* | *appears*] $_{\text{Verb}}$ [*on*] $_{\text{Preposition}}$ [*the green wall*] $_{\text{NP}}$.

(L_4 and K=0): [*The book*] $_{\text{NP}}$ [*on*] $_{\text{Preposition}}$ [*the table*] $_{\text{NP}}$.

(L_5 and K=1): [*The young girl*] $_{\text{NP}}$ [*jumped*] $_{\text{Verb}}$ [*into*] $_{\text{Preposition}}$ [*the deep blue ocean*] $_{\text{NP}}$.

(L_5 and K=0): [*The man*] $_{\text{NP}}$ [*left*] $_{\text{Verb}}$ [*the building*] $_{\text{NP}}$.

(L_6): [*Millersport Highway*] $_{\text{NP}}$ [*crosses*] $_{\text{Verb}}$ [*North French Road*] $_{\text{NP}}$. [130]

At the first sight, the formalization process appears relatively straightforward using these templates. One can simply match these templates against an utterance of interest and annotate noun phrases on the left and right as locatum and relatum, respectively. The union of these spatial languages $\mathbf{L_{SP}} = \{\mathbf{L_4} \cup \mathbf{L_5} \cup \mathbf{L_6}\}$ can address many spatial expressions in English. Nevertheless, $\mathbf{L_{sp}}$ faces three major challenges: syntactic incompleteness, semantic ambiguity, and semantic role ambiguity.

$\mathbf{L_{sp}}$ only covers declarative statements and cannot capture imperative or interrogative statements. As an example, many Human–Robot Interaction (HRI) systems interact in an imperative form and restrict user inputs to the following syntactic frame.

$$\mathbf{L}_7 = \left\{[\text{Command Verb}][\text{NP}][\text{Preposition}][\text{NP}]\right\} \tag{4.7}$$

Moreover, $\mathbf{L_{sp}}$ cannot completely detect spatial patterns stated in the declarative form. Statements such as "*The girl is here*" or "*On the desk, there is a book*" are examples of such utterances. Compound spatial statements are also challenging. These statements can expand to include several spatial relations. "*The book on the table to the right of the laptop which is on top of the notebook close to the pen is mine*." is an example of such nested spatial relations.

Syntactic incompleteness mostly affects the recall measure (i.e., $R = TP \times (TP + FN)^{-1}$ where *TP* denotes true positives and *FN* denotes false negatives) as it is prone to misclassifying a positive example as a false negative. Nevertheless, it can be mitigated by crafting a comprehensive set of accurate patterns to capture the spatial information.

Semantic ambiguity is associated with polysemous behavior (i.e., capacity to have multiple semantically relevant but distinct senses) in both lexical and compositional levels. In the lexical level, this ambiguity is a result of different senses of a given word. As an instance, although in daily English prepositions such as "*on*" and "*under*" frequently appear as spatial indicators, yet they are sometimes used with their non–spatial senses such as following examples.

(1) "*The Church of England has been getting more and more liberal **on** the matter*."
(2) "*They had personal freedoms **under** his rule but they demanded political freedom too."*

Same lexical ambiguity occurs with spatial verbs. There are three verb categories that have some intrinsic spatial meanings including existence verbs (e.g., "*appear"*, "*locate"*, "*remain"*, "*live"*, etc.), motion verbs (e.g., "*jump"*, "*fly"*, "*run"*, "*leave"*, etc.), and trajectory verbs (e.g., "*cross"*, "*connect with"*, "*go along"*, etc.). Like prepositions, not all senses of a spatial verb are spatial. As an example, the motion verb "*leave"* in "*He **left** the office a year ago*" [131] mostly carries temporal information than spatial information. The task of deciding the sense of a word in a given context is referred as word sense disambiguation and is considered as an AI–complete problem [132].

The disambiguation process faces more challenges in compositional semantic ambiguity. In these cases, a word of interest usually has a spatial sense but the overall meaning of an utterance is not referring to a spatial setting. Idioms and metaphors are well–known examples of such situations. Four examples of such idioms are as follows. Examples (1) – (3) are referring to "*passing away"*, *"getting old",* and "*feeling ill"*.

(1) "*He **left** this world a year ago*."

(2) "*Peter is* **over** *the hill.*" [133]

(3) "*She felt* **under** *the weather.*" [133]

(4) "*The thought* **in the back** *of my mind.*" [133]

Semantic ambiguity mostly affects the precision measure (i.e., $P = TP \times (TP + FP)^{-1}$ where *TP* and *FP* denote true positives and false positives, respectively) as it is prone to misclassifying a negative example as a false positive. Contrary to lexical incompleteness, crafting patterns is not a remedy for semantic ambiguity. Hence, machine learning techniques are required to address this challenge.

The third challenge is related to semantic role ambiguity. In standard locative expressions, a locatum appears as a noun phrase followed by a spatial preposition and a noun phrase indicating a pivot and a relatum, respectively. However, this syntactic structure does not cover cases such as having multiple roles for a constituent (e.g., "*The book to the left of laptop on the desk*"), having multiple locatums (e.g., "*a house and a green wall with gate in the background.*"), implicit relatums (e.g., "*a sign saying that plants cannot be picked up on the right.*"), different writing styles (e.g., "*on the table, there is a book*."), and co–references (e.g., "*The book is on the table and the laptop is to its right*").

The process of disambiguating and assigning spatial roles to constituents is referred as spatial role labeling [131] which is a special case of Semantic Role Labeling (SRL) [18]. Automatic annotation of spatial relations is a more challenging task compared to tasks such as automatic recognition of temporal relations. These challenges root in: (1) spatial relations require more complex representation due to their multi–dimensional nature; (2) in contrast to temporal relations, an utterance is not usually ordered by relative spatial arrangements; and (3) spatial event semantics is usually implicit [134].

The mentioned challenges can be solved by learning linguistic regularities and clues. However, there is another type of ambiguity that solving it requires a world model as well. As an example, in our first example, "*Put the laptop on the table to the right of the blue book",* we would guess that the correct interpretation is "*Put the laptop to the right of the book which is on the table*". Another but a less likely interpretation is to "*Put the laptop on the table which is to the right of the book*". This kind of ambiguity cannot be solved by exploiting linguistic regularities but rather requires world knowledge and an inference engine to jointly parse and

disambiguate an utterance [79]. This ambiguity can be solved by examining the existence of objects associated with noun phrases in a world model.

4.1 Related Work

In addition to text–to–scene and text–to–animation conversion systems, automatic extraction of spatial relations from text has been investigated in a few research areas including robotics, geographic information system, and computational linguistics. In Human–Robot Interactions (HRI) spatially–oriented tasks (e.g., navigation, fetching objects) are usually addressed using template matching and then grounding noun phrases using a world model [135]–[137]. Templates for syntactic frames are defined using annotated corpora of user commands [138], [139]. HRI systems mostly focus on direct mapping between a spatial language and a target semantic representation model using a spatial planner [139], [140]. Context predicates [66], Robot Control Language (RCL) [139], CLP(QS) [141] are examples of such representations. Research on spatial relations in geographic information systems is mostly focused on spatial annotation schemes such as Automatic Content Extraction (ACE), Generalized Upper Model (GUM), Geography Markup Language (GML), Keyhole Markup Language (KML), Toponym Resolution Markup Language (TRML) [142], SpatialML [143], and ISO–Space [144].

A systematic approach towards spatial role labeling is studied in [131] which is based on the holistic spatial semantic theory [145] and addresses this task by jointly learning the spatial roles by training a skip–chain Conditional Random Field (CRF) [27] on a set of engineered linguistic features. In [146], Visually Informed Embeddings of Word (VIEW) are introduced to transfer the multimodal background knowledge to this task. These embeddings are extracted using a Long Short–Term Memory (LSTM) network [23] trained on the Microsoft COCO dataset. Results suggest that training a deep feed–forward neural network (DNN) on a combination of these embeddings and features proposed in [131] improves the F_1–score.

A few works are carried out on shared task of spatial role labeling (Task 3) in SemEval2012 [147] and SemEval2013 [148]. In [149], spatial role labeling is formulated as a binary classification task in which some heuristics are introduced to produce a set of candidate <*locatum, pivot, relatum*> triplets. These triplets are used to train a binary SVM classifier to

predict whether a given triplet is a spatial relation or not. In [150], a SVM–HMM sequence classifier is used to classify a sentence word–by–word to a set of possible spatial roles. We also use these share tasks to evaluate our proposed model on spatial role labeling.

Some studies address cognitive–linguistic spatial concepts by constructing spatial ontologies. A knowledgebase approach is studied in [151] which propose an ontological semantics for spatial expressions that support computational processing. It extends Generalized Upper Model (GUM) [152] to produce a highly detailed extension for the spatial domain. A spatial ontology population scheme is proposed in [153] which utilizes structured machine learning to populate a spatial ontology with two layers: spatial role labeling layer and spatial qualitative labeling layer. The first layer extracts a set of spatial signals from a given sentence and the second layer represents these signals based on qualitative spatial representation models. In [154], [155] an RBF–SVM classifier is trained on a combination of lexical and semantic features to automatically infer temporally–anchored spatial knowledge (i.e., where an entity is not located and for how long).

Disambiguating spatial prepositions is also investigated in the literature. In [133], a fast disambiguation scheme is proposed that utilizes a few WordNet–based heuristics. This scheme can recognize the most common metaphoric uses of prepositions, and abstract locatums and relatums. In [131], a few linguistic features are used to train a Naive Bayes and a maximum entropy classifiers to disambiguate spatial prepositions. In [156], preposition sense disambiguation is addressed using a classification rule discovery scheme, whereas in [157], this task is addressed using semantic role resources such as WordNet [19], FrameNet [20], and OpenCyc [158]. We extract a few semantic features using WordNet [19]. In [159], it is shown that joint learning of senses and semantic roles of prepositional phrase can enhance the accuracy. A few work such as [160] and [161] are carried out on a shared task of proposition disambiguation in SemEval2007 [127].

4.2 Approach

Given an arbitrary utterance, we first investigate whether it carries spatial information or not. And if it does, we identify and label those signals. The first task is a word–sense disambiguation task, whereas the second task is related to spatial role labeling. To address the

first task, we formulate it as a binary classification problem as follows. Given a sentence of length L, a set of potential pivots (i.e., spatial prepositions and spatial verbs) are detected using a dictionary. A context window size of $2k+1$ (k is determined using sensitivity analysis) is then centered on a potential pivot (i.e., k words to the left and k words to the right of a pivot) and a set of corresponding features are extracted. These features are then fed to a deep Convolutional Neural Network (CNN) to predict whether a candidate pivot implies a spatial sense or not. The architecture of this model is shown in Figure 4.1. It consists of a few convolutional layers each followed by a max–pooling layer. Outputs of these layers are fed to a fully–connected layer followed by a soft–max layer to predict the probability of two classes (0: not spatial and 1: spatial). Considering the complexity of the datasets, we fix the number of convolutional and fully–connected layers to two and optimize the number of neurons in fully–connected layers. Experimental results discussed in section 4.6 show that this architecture achieves excellent results.

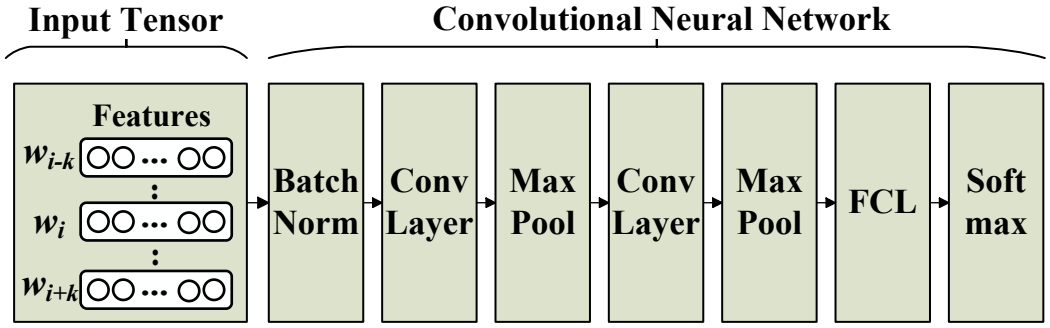

Figure 4.1. The convolutional neural model utilized for word–sense disambiguation of spatial pivots. Input is a set of linguistic features extracted from a context window, and output is either 0 denoting that a pivot is not referring to a spatial sense or 1 denoting otherwise

The second task is formulated as a sequence–to–sequence classification task which maps a word in a sentence to one of five classes (0: not a spatial signal, 1: locatum, 2: relatum, 3: pivot, 4: both relatum and locatum). We approach this task by tagging a sentence word–by–word. For all words within a sentence starting from the leftmost word, a context window size of $2k+1$ is centered on a current word and some features are extracted from that window. These features along with prediction from a preceding word are then fed to a deep CNN as illustrated in

Figure 4.2. It is shown that using previous predictions is an effective strategy in designing CNN models [162]. As an example, in the POS–tagging task given that the previous prediction is an adjective, there is a high chance that current word is an adjective or a noun. Finally, each detected pivot combined with its immediate preceding and proceeding locatum and relatum is used to map detected spatial roles to a predicate–argument structure. We also propose a hybrid feature that integrates a set of linguistic features with universal word vectors. Because universal embeddings convey information regarding similarities and dissimilarities among words, integrating them with local information from a corpus can enhance the performance.

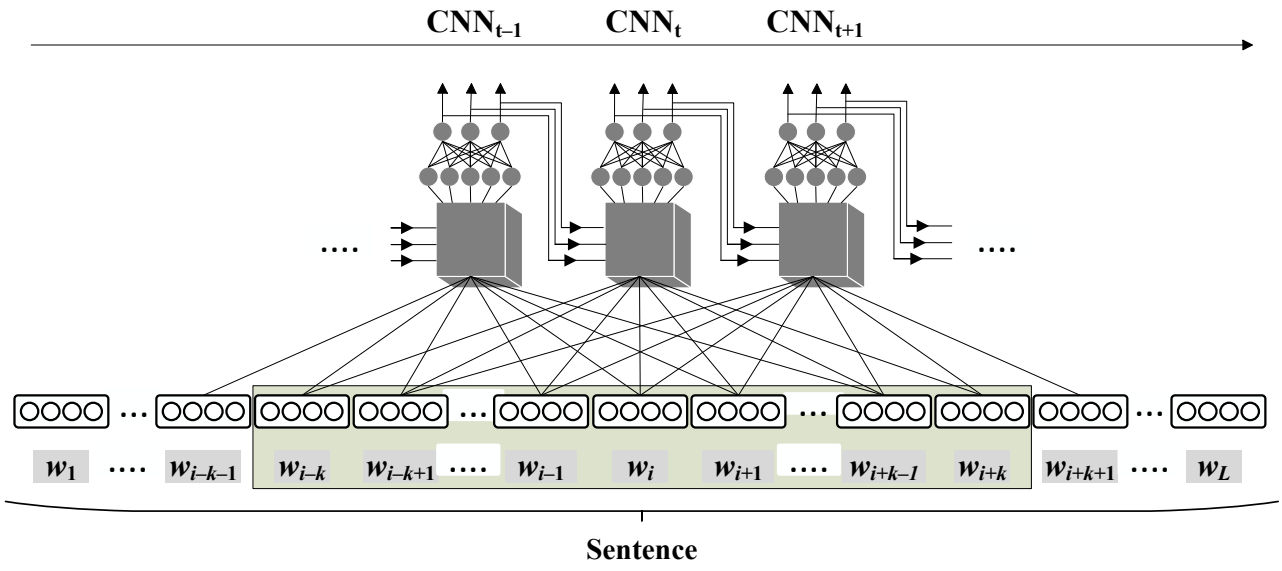

Figure 4.2. The proposed CNN architecture for spatial role labeling. Input consists of linguistic features of each word and predictions from previous words and output is one of the following classes: 0: not a spatial signal, 1: locatum, 2: relatum, 3: pivot, 4: both relatum and locatum.

4.3 Features

We investigate four feature sets to decide a final feature representation. These feature sets include engineered linguistic features, pre–trained word embeddings, corpus–trained word embeddings, and proposed hybrid features. In most of the related work, either local engineered features (i.e., linguistic features) or word vectors are utilized. We propose hybrid feature sets that

benefit from both engineered features and word vectors. An example of hybrid feature extracted from a simple sentence is shown in Appendix G.

4.3.1 Linguistic Features

Linguistic features consist of thirteen engineered features including five lexical features, four syntactic features, and four semantic features. For a given symmetric context window (within the text) W: $[w_{i-k}, ..., w_i, ..., w_{i+k}]$ of size $2k+1$ centered on a word of interest (i.e., k words to the left and k words to the right where k is determined using sensitivity analysis), these features are as follows.

Lexical Features

1. Unigrams:

$$\mathbf{\Phi_1}: [w_{i-k}, ..., w_i, ..., w_{i+k}] \tag{4.8}$$

2. Lemmas:

$$\mathbf{\Phi_2}: [l_{i-k}, ..., l_i, ..., l_{i+k}] \tag{4.9}$$

3. 1–skip–bigrams:

$$\mathbf{\Phi_3}: [w_{i-k}\ w_{i-k+1}, w_{i-k}\ w_{i-k+2}, ..., w_i\ w_{i+1}, ..., w_{i+k-1}\ w_{i+k}] \tag{4.10}$$

4. Probability of unigrams belonging to class c:

$$\mathbf{\Phi_4}: [p_c(w_{i-k}), ..., p_c(w_i), ..., p_c(w_{i+k})] \tag{4.11}$$

5. Probability of 1–skip–bigrams belonging to class c:

$$\mathbf{\Phi_5}: [p_c(w_{i-k+1}|w_{i-k}), ..., p_c(w_{i+1}|w_i), ..., p_c(w_{i+k}|\ w_{i+k-1})] \tag{4.12}$$

Unigrams and lemmas are extracted using CoreNLP toolkit [29] by tokenizing and lemmatizing a sample data, and 1–skip–bigrams are extracted using NLTK toolkit [163]. Probabilities are computed using Maximum Likelihood Estimation (MLE) considering the occurrences of unigrams and skip–grams within a training set.

Syntactic Features

6. Word–level POS–tags:

$\mathbf{\Phi_6}$: $[pos_w(w_{i-k}), \ldots, pos_w(w_i), \ldots, pos_w(w_{i+k})]$ (4.13)

7. Phrase–level POS–tags of words:

$\mathbf{\Phi 7}$: $[pos_p(w_{i-k}), \ldots, pos_p(w_i), \ldots, pos_p(w_{i+k})]$ (4.14)

8. Phrase–level POS–tags of immediate ancestors of words:

$\mathbf{\Phi_8}$: $[pos_a(w_{i-k}), \ldots, pos_a(w_i), \ldots, pos_a(w_{i+k})]$ (4.15)

9. Minimum syntactic distance between words in a context window and a center word:

$\mathbf{\Phi_9}$: $[dis(w_{i-k},w_i),\ldots,0,\ldots,dis(w_i,w_{i+k})]$, $dis(w_i,w_i)=0$ (4.16)

Word–level and phrase–level POS–tags are extracted using CoreNLP POS–tagger [30] and syntactic parser [31], respectively. For a given pair of words (w_1,w_2), the minimum syntactic feature is computed as follow.

$$dis(w_1, w_2) = d(w_1, r) + d(w_2, r) - 2 \times d(lca(w_1, w_2), r) \quad (4.17)$$

r denotes the root of a parse tree, $d(w,r)$ is a distance between the root and a word, and $lca(w_1,w_2)$ is the lowest common ancestor of words w_1 and w_2.

Semantic Features

10. Collapsed dependencies between a center word and words within a context window:

$\mathbf{\Phi_{10}}$: $[dep(w_{i-k},w_i),\ldots,None,\ldots,dep(w_i,w_{i+k})]$, $dis(w_i,w_i)=None$ (4.18)

11. Named–entities:

$\mathbf{\Phi_{11}}$: $[ne(w_{i-k}), \ldots, ne(w_i), \ldots, ne(w_{i+k})]$ (4.19)

12. An indicator to decide whether a word refers to an abstract or a physical entity:

$\mathbf{\Phi_{12}}$: $[phy(w_{i-k}), \ldots, phy(w_i), \ldots, phy(w_{i+k})]$ (4.20)

13. Polysemy score of words:

$\mathbf{\Phi_{13}}$: $[pol(w_{i-k}), \ldots, pol(w_i), \ldots, pol(w_{i+k})]$ (4.21)

The Stanford named–entity recognizer [35] and the universal dependency parser [36] are utilized to extract $\mathbf{\Phi_{10}}$ and $\mathbf{\Phi_{11}}$. In case that a word or a gram is not a named entity, it is marked as *none*. $\mathbf{\Phi_{12}}$ is extracted as follows. First, all nouns are extracted and searched in WordNet.

Found nouns that are children of the *physical entity* synset (i.e., hypernym hierarchy of at least one of their senses includes *physical entity*) are marked as *physical*. Those nouns whose hypernym hierarchy of their all senses are children of the *abstract entity* are tagged as *abstract*. Words missing from WordNet are tagged as *none*. Also, a pairwise logical OR is performed on proper nouns captured by the named–entity recognizer and the search results from WordNet to augment this feature. The polysemy score of a word w is computed as $C(w)^{-1}$, where $C(w)$ is the number of senses of w in WordNet.

Finally, the linguistic feature set ($\mathbf{F_1}$) is defined as a concatenation of features $\mathbf{\Phi_1}$–$\mathbf{\Phi_{13}}$ as follows.

$$\mathbf{F_1}: [\mathbf{\Phi_1}\, \mathbf{\Phi_2}\, \mathbf{\Phi_3}\, \mathbf{\Phi_4}\, \mathbf{\Phi_5}\, \mathbf{\Phi_6}\, \mathbf{\Phi_7}\, \mathbf{\Phi_8}\, \mathbf{\Phi_9}\, \mathbf{\Phi_{10}}\, \mathbf{\Phi_{11}}\, \mathbf{\Phi_{12}}\, \mathbf{\Phi_{13}}]^{T} \tag{4.22}$$

4.3.2 Universal Word Embeddings

Word2Vec Embeddings

Word2Vec pre–trained vectors are trained on part of Google News dataset with about 100 billion tokens. The model contains 300–dimensional vectors for 3 million words and phrases and is publicly available. Given a context window W: $[w_{i-k}, \ldots, w_i, \ldots, w_{i+k}]$, we define a Word2Vec representation of a window as follows.

$$\mathbf{\Phi_{14}}: [H_{w2v}(w_{i-k}), \ldots, H_{w2v}(w_i), \ldots, H_{w2v}(w_{i+k})] \tag{4.23}$$

$H_{w2v}(w_k)$ is a hash function (dictionary) that retrieves a pre–trained vector of a given word (w_k). For a bigram $[w_i, w_{i+1}]$, we represent both constituent words with the corresponding vector of the bigram: $[H_{w2v}(w_i) = H_{w2v}(w_i\ w_{i+1}), H_{w2v}(w_{i+1}) = H_{w2v}(w_i\ w_{i+1})]$. We use a similar approach to address the trigrams.

GloVe Embeddings

GloVe embeddings are trained on documents collected by Common Crawl with 840 billion tokens. The model contains 300–dimensional vectors for 2.2 million words which are publicly available. Similarly, given a context window W: $[w_{i-k}, \ldots, w_i, \ldots, w_{i+k}]$, we define a GloVe feature as follows:

$$\mathbf{\Phi_{15}}: [H_G(w_{i-k}), \ldots, H_G(w_i), \ldots, H_G(w_{i+k})] \tag{4.24}$$

$\underline{H_G}(w_k)$ is a hash function of word–vector pairs.

In both word embedding models, we use a dictionary implemented as a hash function containing <*key*, *value*> pairs to retrieve a word vector (value) of a given skip–gram (key). In case that a skip–gram is found in this dictionary, we assign all the words within the skip–gram with the same word vector.

Local Word Embeddings

In addition to pre–trained Word2Vec vectors, local word embeddings are trained on a corpus based on SGNS model and using genism semantic modeling library [164]. For a context window W:$[w_{i-k}, \ldots, w_i, \ldots, w_{i+k}]$, these local word embeddings are defined as follows.

$$\mathbf{\Phi_{16}}: [H_{L\text{-}w2v}(w_{i-k}), \ldots, H_{L\text{-}w2v}(w_i), \ldots, H_{L\text{-}w2v}(w_{i+k})] \quad (4.25)$$

$H_{L\text{-}w2v}(w_k)$ is a hash mapping between words and locally–trained word vectors.

4.3.3 Hybrid Features

Pre–trained word vectors represent universal and distributional semantics whereas engineered features represent local discriminative signals. To benefit from both, we define hybrid features as the amalgamation of word embeddings and linguistic features. For this purpose, we replace the linguistic features $\mathbf{\Phi_1}$–$\mathbf{\Phi_3}$ and $\mathbf{\Phi_5}$ with corresponding word embeddings. Two hybrid features are defined as follows.

$$\mathbf{F_{H\text{-}W2V}}: [\mathbf{\Phi_{14}\, \Phi_4\, \Phi_6\, \Phi_7\, \Phi_8\, \Phi_9\, \Phi_{10}\, \Phi_{11}\, \Phi_{12}\, \Phi_{13}}]^T \quad (4.26)$$

$$\mathbf{F_{H\text{-}GloVe}}: [\mathbf{\Phi_{15}\, \Phi_4\, \Phi_6\, \Phi_7\, \Phi_8\, \Phi_9\, \Phi_{10}\, \Phi_{11}\, \Phi_{12}\, \Phi_{13}}]^T \quad (4.27)$$

4.4 Datasets

4.4.1 PDEP Dataset

The Pattern Dictionary of English Prepositions (PDEP) [165] is a publicly available lexical resource collected as a part of The Preposition Project (TPP) for the study of preposition behavior. This repository contains example sentences drawn from three corpora including FrameNet, Oxford English Corpus, and British National Corpus. PDEP contains 82,329 annotated example sentences of 1,061 senses under 304 prepositions [165]. Prepositions are

classified into twelve classes including activity, agent, backdrop, cause, membership, exception, scalar, spatial, tandem, temporal, topic, and tributary preposition classes. It provides senses of a preposition, its pattern, substitutable prepositions, syntactic position, semantic class, super–sense, cluster, relation, and primary implicature.

The spatial class contains 169 senses under 78 spatial prepositions and 19,413 positive annotated examples. To compile the dataset, we extracted all members of the spatial class (i.e., prepositions that have at least one spatial sense) and filtered out 36 spatial prepositions including archaic prepositions (e.g., "*betwixt"*, "*nigh"* and "*thwart"*), technical prepositions (e.g., "*abaft"*), limited dialect prepositions (e.g., "*outwith"*), and prepositions with less than ten examples (e.g., "*fore"* and "*sans"*). This resulted in a corpus of 73 spatial prepositions and 596 senses out of which 169 are spatial. Among them, *upon* with 25 (8 spatial and 17 non–spatial) different senses has the highest polysemy. *Aboard* and *midst* are the only prepositions whose all senses are spatial, and *by* has the minimum ratio of spatial senses (2 spatial and 21 non–spatial). The final corpus consists of 19,103 positive instances and 24,026 negative instances. It is an almost balanced dataset consisting of 43,129 samples with skewness of 0.2289 and Kurtosis measure of −1.9473 and consists of 1,549,492 tokens and 59,814 words. We further split the dataset to a train set with 39,000 samples and a test set with 4,129 samples (2,000 spatial and 2,129 non–spatial samples). This dataset is used for the task of word–sense disambiguation of spatial prepositions (binary word–sense disambiguation: spatial or not–spatial).

4.4.2 IAPR Dataset

This dataset is a subset of CLEF IAPR TC–12 image benchmark [166] and contains descriptions of 613 images taken by tourists. The descriptions describe objects and their absolute and relative positions within the images. These descriptions are expressed within 1,213 sentences with an average length of 15 words and standard deviation of 8. The corpus consists of 20,027 terms and 1,257 words. This dataset is provided for the shared task on spatial role labeling (task 3) in Semantic Evaluation (SemEval) workshop 2012 and 2013 [147], [148]. In SemEval–2012, spatial roles are assigned to the headwords of constituents whereas in SemEval–2013 annotations are span–based. The goal of this task is to extract a set of static spatial relations represented as spatial relation triplets: <*trajector, indicator, landmark*> and then to classify the relation type to

one of the coarse–grained region, direction, and distance types. A sample annotates sentence in this corpus is as follows:

<*SENTENCE* id='s605'>
<*CONTENT*>blue sky in the background .</CONTENT>
<*TRAJECTOR* id='tw1'> sky</TRAJECTOR>
<*LANDMARK* id='lw6'>undefined</LANDMARK>
<*SPATIAL_INDICATOR* id='sw2'>in </SPATIAL_INDICATOR>
<*RELATION* id='r0' sp='sw2' tr='tw1' lm='lw6' general_type='region'/>
</*SENTENCE*>

The statistics of this dataset are shown in Table 4.1.

Table 4.1. The statistics of IAPR dataset

	#sentences	**#Terms**	**#Words**	**#Relatums**	**#Locatums**	**#Pivots**	**#Relations**
Train	600	9,048	836	716	661	670	766
Test	613	10,979	778	872	743	796	940
Total	1,213	20,027	1,257	1,593	1,408	1,464	1,715

4.4.3 Scene Dataset

The scene dataset is collected as a part of our efforts to learn and ground the spatial relations. This dataset is collected by recruiting annotators on Amazon Mechanical Turk (AMT). We presented an annotator with a sample scene of a 3D environment and asked her to describe spatial relations within the scene in the form of locative expressions. For each task, the annotator is provided with a snapshot of a 3D scene and a set of object names within the scene, and is asked to provide at least 20 sentences consisting of at least 5 distinct spatial prepositions. Annotators are provided with 196 snapshots of 132 scenes (each scene is annotated by only one annotator). The scenes are collected from the Stanford Scene Database [74]. The collected corpus contains 3,920 sentences consisting of 37,836 terms and 987 words. The annotators have described the spatial relations using 27 distinct spatial prepositions. We further split the collected

dataset to a train set and a test set. The train set consists of 3,000 sentences whereas the test set consists of 920 sentences. We will further elaborate on this dataset in section 5.4.

4.5 Learning Models

To have a solid baseline on both tasks, we utilize linear, ensemble, and deep models. Linear models include a logistic regression classifier, a k–nearest neighbor (K–NN) classifier, a Bernoulli naïve Bayes classifier, a linear support vector machines (SVM), and ensemble classifiers include a random forest classifier, an Ada–boost classifier, and a bagging classifier. These classifiers are applied using the scikit–learn machine learning library [167]. We also use two deep learning models including a fully–connected feed–forward deep neural network (DNN) and a deep convolutional neural network (CNN) (shown in Figures 4.1 and 4.2) using the TensorFlow library [168]. For the word–sense disambiguation task, models are trained on a surrounding context of a candidate pivot, whereas for spatial role labeling task, models are trained on a surrounding context of each and every word within a sentence, while utilizing previous predictions.

4.6 Experimental Setup

4.6.1 Preprocessing

A few regular expressions are used to replace the numbers with <*NUM*> symbol, pad a context window by <*PAD*> symbol, and remove all characters except alphabetic characters and the punctuation marks. Preprocessing of features is as follows. For universal word embeddings, missing words in pre–trained models are generated randomly by sampling each dimension from U~[–1,+1]. We also reduce the dimensionality of pre–trained vectors from 300 to 10, 50, 100, and 200 using Principal Component Analysis (PCA) to investigate the effect of different dimensions of word vectors on models. For locally–trained word embeddings, we do not perform any feature preprocessing. For linguistic features, numeric identifiers of POS–tags, unigrams, 1–skip–bigrams, dependencies, named–entities, and probability vectors are mapped to the range of [–1,+1] using min–max normalization. For each feature set, context windows of sizes 3, 5, 7, …,

19, and 21 are considered (e.g., a window size of 11 contains 5 words to the right and 5 words to the left of a center word).

4.6.2 Model Setup

Both the DNN and the CNN models are trained using Adam stochastic optimizer [169] with a learning rate of 1E−4 over the mini–batches of size 250. The mini–batches are uniformly sampled with replacement from the training set. Both models utilize a cross–entropy loss function with one–hot output representation. The models use dropout regularization [170] with a probability of p=0.5 and batch normalization [171] on input layer. Both models also utilize Rectified Linear Units (ReLU) in hidden layers and a softmax function in the output layer.

The DNN model is defined as a four–layer network (3 hidden layers + softmax layer). The CNN architecture consists of two convolutional layers each followed by a max–pooling layer and two fully–connected layers. The probability of classes is computed using a softmax layer. We fixed the number of epochs to 50,000 and used 32, 64, and 128 filters with sizes of 2×5, 3×5, and 5×5 with a stride of 1. We also used windows of sizes 1×3, 2×2, 3×2 and 3×5 for pooling. Sizes of hidden layers are considered as hyper–parameters and are optimized using a random search. For this purpose, 10% of the training set is sampled as the validation set. For different feature sets, these sizes are set to 500, 800, and 1,000 neurons for DNN model and 400 and 800 neurons for fully–connected layers of CNN model.

The random search strategy is also employed to optimize a set of hyper–parameters of linear and ensemble models. These models also utilize an L2 regularization to avoid over–fitting. We repeat the search for 200 times and select the best model based on the performance on the validation set. The hyper–parameters of these models are discussed in Appendix C.

4.7 Experimental Results

4.7.1 Disambiguating Spatial Prepositions

This task is evaluated on PDEP Dataset. We first investigate the effect of context window size and dimensions of word embeddings on the F_1–score. These effects for DNN model are shown in Figures 4.3 and 4.4, respectively. Other models show similar behavior as well.

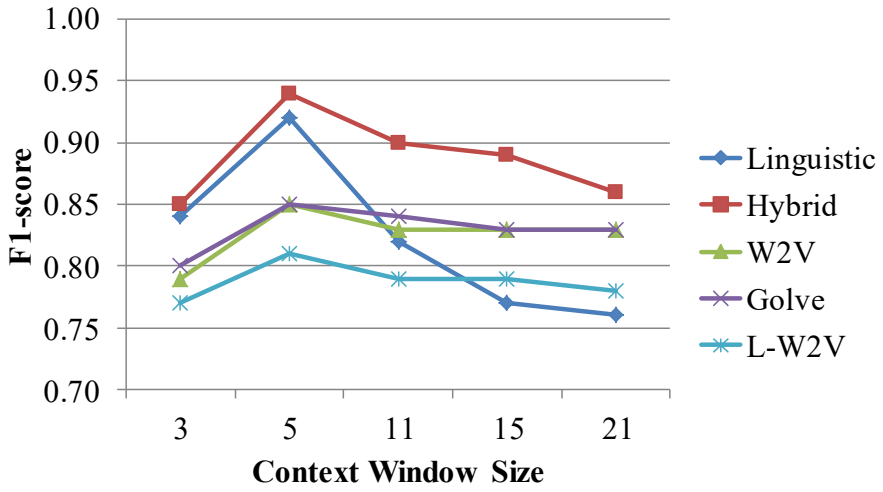

Figure 4.3. The effect of context window size on F_1–score of DNN model in word–sense disambiguation task. The best F_1–score is achieved with window size of *k*=5

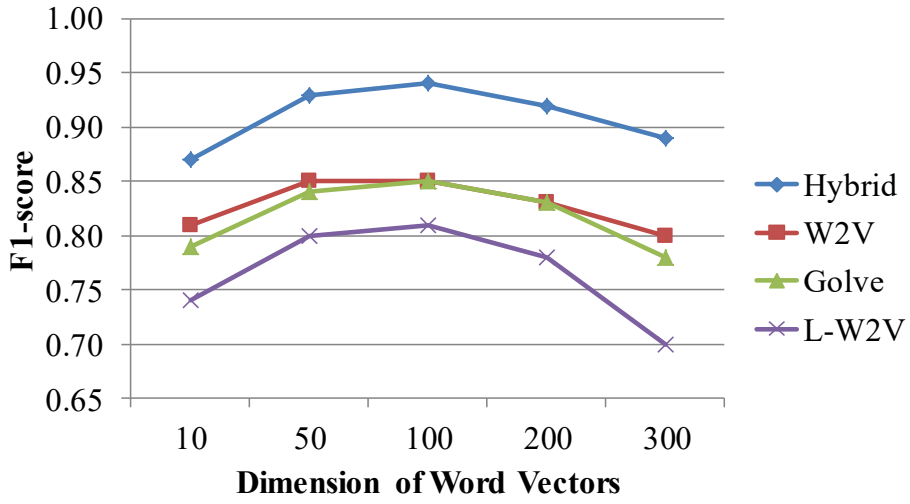

Figure 4.4. The effect of word vector size on F_1–score of DNN model in word-sense disambiguation task. The best F_1–score is achieved with word embeddings of size 100

As illustrated in Figure 4.3, as window size increases, F_1–score decreases proportionally. This suggests that senses of a preposition only depend on a small local context window. This is because a dependent and a governor of a preposition which provide the clues about its sense tend to appear very close to a preposition (i.e., preceding and proceeding noun phrases). Also, it is shown that the best F_1–score is achieved with a window size of 5 which implies that a smaller window misses useful information whereas larger windows introduce higher noise to signal ratio.

Also, as shown in Figure 4.4, the best F_1–score is achieved by reducing word vector dimensions from 300 to 100. It is noteworthy that an optimal dimension size depends on the size of corpus.

We also analyze the accuracy and F_1–score of different learning models with respect to the introduced feature sets. Note that, because the preposition dataset is almost balanced, accuracy is a valid measure. Classification results of learning models trained using word vectors are shown in Table 4.2, and results for linguistic and hybrid features are shown in Table 4.3. The context window size and the dimension of word vectors are fixed to 5 and 100, respectively.

Table 4.2. Classification performance of learning models on a context window size of 5 with three word vector models of size 100 in word-sense disambiguation task. The best F_1–score is achieved by training the proposed CNN model with pre–trained W2V word embeddings

Classifier	**pre–trained W2V**		**pre–trained GloVe**		**locally–trained W2V**	
	F_1–score	Accuracy	F_1–score	Accuracy	F_1–score	Accuracy
Ada–boost	0.7824	79.44%	0.7825	79.73%	0.7866	77.19%
Random forest	0.7788	80.36%	0.8089	82.97%	0.7758	79.87%
Bagging	0.8020	82.08%	0.8050	82.27%	0.7785	80.24%
Linear SVM	0.7929	80.67%	0.8012	81.50%	0.7758	79.87%
K–NN	0.7720	80.43%	0.8009	82.44%	0.7166	74.64%
Logistic regression	0.8195	83.02%	0.8234	83.34%	0.7615	77.74%
DNN	0.8410	84.96%	**0.8414**	**85.20%**	**0.7996**	**81.06%**
CNN	**0.8425**	**85.08%**	0.8401	84.82%	0.7955	80.60%

Table 4.3. Classification performance of learning models with linguistic and proposed hybrid features on a context window of size 5. The proposed CNN model trained on the proposed hybrid feature outperforms other feature–model combinations

Classifier	**Linguistic**		**Hybrid**	
	F_1–score	Accuracy	F_1–score	Accuracy
Ada–boost	0.9279	93.19%	0.9330	93.58%
Random forest	0.9147	91.96%	0.9302	93.46%
Bagging	0.9223	92.66%	0.9307	93.46%
Linear SVM	0.8909	89.66%	0.9317	93.51%
K–NN	0.9070	91.31%	0.8817	89.15%
Naïve Bayes	0.9018	90.80%	———	———
Logistic regression	0.9141	91.84%	0.9321	93.51%
DNN	0.9265	93.23%	0.9388	94.16%
CNN	**0.9312**	**93.54%**	**0.9398**	**94.21%**

The following observations can be pointed out:

(1) Pre–trained embeddings outperform locally–trained embeddings in terms of accuracy and F_1–score by 4.85% and 5.39%, respectively. This implies that the PDEP corpus is not big enough to train local word embeddings.

(2) Pre–trained GloVe model slightly outperform pre–trained Word2Vec model when used with linear and ensemble classifiers. On the other hand, Word2Vec model achieves a better F_1–score when used with DNN. As shown in Figures 4.3 and 4.4, both pre–trained models demonstrate a similar behavior.

(3) Feature sets that exploit corpus information (i.e., linguistic and hybrid features) outperform universal features. Also, it is shown that combining information from both local corpus and universal word distributions results in the best performance.

(5) Regardless of classifier type, the proposed hybrid feature outperforms other feature sets in terms of accuracy and F_1–score. Because universal embeddings convey information regarding the similarities and dissimilarities among words, integrating them with information from a local corpus enhances the performance.

(6) On average, deep learning models outperform the linear and ensemble classifiers, and ensemble models outperform the linear models.

(7) Best performance is achieved by extracting hybrid features from a context window of size 5 and word vectors of size 100 and feeding it to a CNN with two convolution layers. This model achieves accuracy and F_1–score of 94.21% and 0.9398, respectively, which is an improvement of 7.06% over [131].

(8) Error analysis suggests that in most of the feature–model combinations, on average 40% of prediction errors are false negatives and 60% are false positives. This is due to a slight skewness in the dataset (i.e., the skewness of 0.2289 and Kurtosis measure of –1.9473).

(9) Error Analysis also suggests that the distribution of error over the prepositions in fine–grained disambiguation is proportional to the sample size. For example, *upside* with only 17 training and 3 test examples has the worst accuracy (77.77%), whereas *over* with 946 training and 82 test examples has the best accuracy (98.8%). Because the task is formulated as a binary classification task, the number of senses does not affect the accuracy.

(10) Considering the sparsity of natural language and the fact that idioms are very rare events within a language, some sample idioms that only appear in test set result in

misclassification. For example, prepositions within the following idioms are misclassified as spatial:

"*If it's a good day I feel* ***on top of*** *the world.*"

"*Try to avoid flitting* ***from*** *academic twig to twig.*"

We furthermore test the statistical significance of the classification results using McNemar's test. The results shown in Table 4.4 suggest that the classification results are statistically significant.

Table 4.4. Results of the McNemar's statistical test on the classification results of the spatial preposition disambiguation

Classifier Pair	χ^2 estimation	p
CNN ⊗ Ada–boost	14.087	<0.005
CNN ⊗ Random forest	19.360	<0.005
CNN ⊗ Bagging	9.375	<0.005
CNN ⊗ Linear SVM	21.807	<0.005
CNN ⊗ K–NN	24.300	<0.005
CNN ⊗ Logistic regression	18.241	<0.005
CNN ⊗ DNN	3.848	<0.05

4.7.2 Spatial Role Labeling

This task is evaluated on IAPR and scene datasets. Neither of these datasets is balanced and hence we use precession, recall, and F1–scores to evaluate the models. Labeling the spatial roles within the scene dataset is simpler than the IAPR dataset as its samples are stated in form of locative expressions. Therefore, as a baseline, we assign spatial roles to the dataset using a Conditional Random Field (CRF) sequence classifier and linguistic feature set. CRF classifier is implemented using the Mallet toolkit [172]. The results are shown in Table 4.5. As shown, due to the simplicity of this dataset, the CRF model achieves an excellent performance on the test set.

Table 4.5. The performance of a CRF model with the linguistic feature set on the scene dataset. Because all samples are in form of locative expressions, the learning model achieves a high F_1–score

Role	Precision	Recall	F_1–score
Locatum	0.9922	0.9989	0.9959
Relatum	0.9877	0.9837	0.9857
Pivot	0.9846	0.9983	0.9914

The IAPR dataset, on the other hand, is more challenging as it is in an unrestricted form and contains samples with implicit relatums and compound relations. First, we investigate the effect of context window size and embedding dimensions on this dataset. These effects on a bagging classifier with the linguistic feature set and Word2Vec pre–trained features are shown in Figures 4.5 and 4.6, respectively. We observed a similar behavior by other learning models.

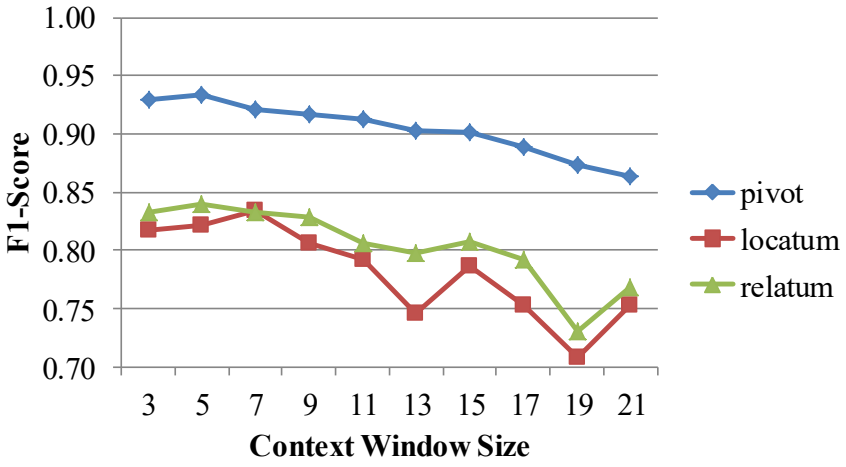

Figure 4.5. Effect of the context window size on F_1–score of a bagging classifier trained with linguistic features. Role labeling of pivots and relatums reaches the highest F_1–score with context window size of 5, whereas for locatums the best results are achieved with a context window of size 7. The final context window size is set to 7

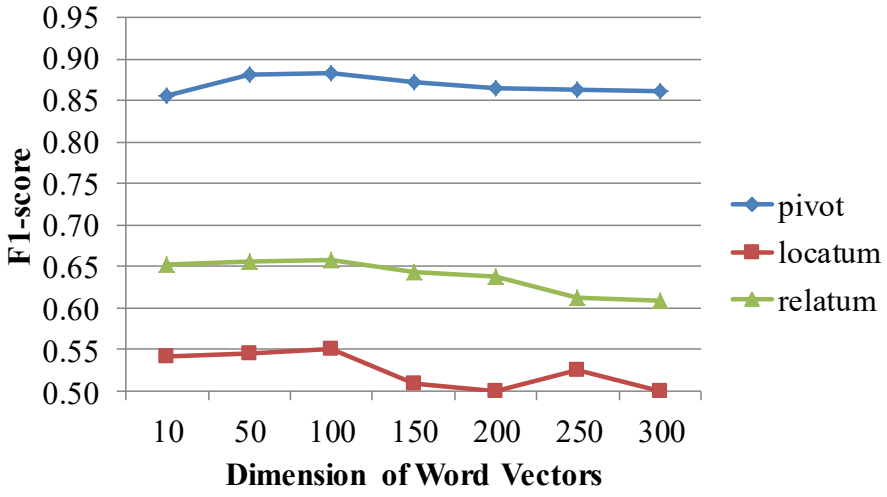

Figure 4.6. Effect of the dimension of word vectors on F_1–score of a bagging classifier trained with linguistic features. Word vectors with a size of 100 results in the best performance for all three spatial roles.

As shown in Figure 4.5, the best results for pivot (similar to disambiguation task) and relatum labeling are achieved with a window size of 5. However, the best results for locatum classification are achieved with a window size of 7. This indicates that a locatum depend on a wider context window in comparison with a spatial pivot and a relatum. Furthermore, as shown in Figure 4.6, word vectors with a dimension of 100 achieve the best performance for all three spatial roles. Given these observations, we clamp the window size and word dimension to 7 and 100, respectively.

We also investigate the effect of feature sets on prediction quality. We train a set of ensemble and linear models (i.e., Ada–boost, bagging, random forest, RBF–SVM, K–NN, and logistic regression) using the four introduced feature sets and average the F_1–score. Results are shown in Figure 4.7. As shown, the proposed hybrid feature outperforms other features in all three classes. Finally, we compare the performance of the proposed CNN model (Figure 4.2) with the baseline results provided by organizers and three participants in SemEval–2012 and SemEval–2013. The baseline provided by organizers is achieved using a Conditional Random Field (CRF) classifier. Participants utilize a SVM–HMM and multi–class SVM models. Results on spatial role labeling for spatial pivots, locatums, and relatums are shown in Tables 4.6, 4.7, and 4.8, respectively.

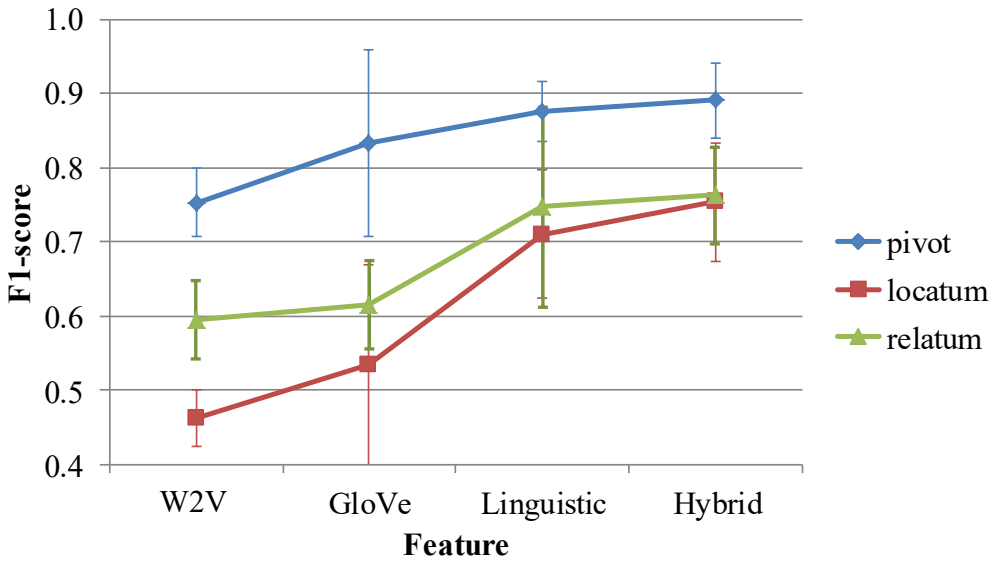

Figure 4.7. The effect of different feature sets on F_1–score. This score is computed by averaging the F_1–score of three ensemble and four linear models. Our proposed hybrid feature outperforms other feature sets.

Table 4.6. Results of predicting spatial pivots. Our proposed CNN model outperforms the state-of-the-art models in terms of recall and F1-score

Classifier	Precision	Recall	F_1–score
Skip–Chain CRF [147]	0.913	0.887	0.900
Multi–Class SVM–1 [149]	0.928	0.712	0.806
Multi–Class SVM–2 [149]	0.940	0.732	0.823
SVM–HMM–1 [150]	0.967	0.889	0.926
SVM–HMM–2 [150]	**0.968**	0.585	0.729
Proposed CNN	0.931	**0.935**	**0.933**

Table 4.7. Results of predicting locatums. Proposed CNN outperforms other models with a significant difference in terms of precision, recall, and F_1-score (21.50% improvement)

Classifier	Precision	Recall	F_1–score
Skip–Chain CRF [147]	0.697	0.603	0.646
Multi–Class SVM–1 [149]	0.731	0.621	0.672
Multi–Class SVM–2 [149]	0.782	0.646	0.707
SVM–HMM–1 [150]	0.684	0.681	0.682
SVM–HMM–2 [150]	0.682	0.493	0.572
Proposed CNN	**0.899**	**0.823**	**0.859**

Table 4.8. Results of predicting relatums. Proposed CNN combined with proposed hybrid feature outperforms other models in terms of recall and F_1-score

Classifier	Precision	Recall	F_1–score
Skip–Chain CRF [147]	0.773	0.740	0.756
Multi–Class SVM–1 [149]	0.871	0.645	0.741
Multi–Class SVM–2 [149]	**0.894**	0.680	0.772
SVM–HMM–1 [150]	0.741	0.835	0.785
SVM–HMM–2 [150]	0.801	0.560	0.659
Proposed CNN	0.883	**0.876**	**0.879**

As shown, our proposed CNN model combined with proposed hybrid feature enhances F_1–score measures of predicting spatial pivots, locatums, and relatums by 0.07 (7.56%), 0.152 (21.50%), and 0.094 (11.97%), respectively. In comparison with relatum and locatum predictions, the spatial pivot has the least improvement. This implies that spatial roles of preceding words of a pivot do not affect it significantly and thus feeding a CNN with its preceding predictions does not enhance its performance.

We addressed two tasks regarding spatial expressions in natural language including word–sense disambiguation of spatial prepositions and semantic role labeling of spatial signals within unrestricted utterances. We also used a set of features to investigate the effect of features on these tasks. Results suggested that in both tasks deep convolutional models outperform other learning models. Also, results show that when these models are trained using a set of universal and local features, they achieve their best performance.

Chapter 5. Inferring and Grounding Spatial Relations

In this chapter, we address the spatial relations in symbolic and visual spaces. We first investigate these relations in language (symbolic) and how to map them to a visual space (world). We then analyze the spatial relations in the world model and address the mapping from this space to language. Finally, we develop a recommendation scheme to model the co–occurrences among different objects.

Spatial relations between objects are important for various tasks in robotics [173], machine vision [174], language processing [131], computer graphics [5], and geographic information systems [175]. Spatial knowledge can be represented by both spatial language in a symbolic space and geometric representation in a visual space. Although humans can map and process these two linguistic–visual spaces with minimum effort, modeling this mapping is a challenging task for software. The process of coding and decoding spatial knowledge between these two spaces is referred as inferring and grounding the spatial relations.

As shown in Figure 5.1, inferring the spatial relations is a mapping from a world model to a language space whereas grounding the spatial relations is a mapping from a language space to a world model. Grounding the spatial relations is useful for world manipulation and motor control scenarios (e.g., an agent is instructed to position an object in relation to another object). Inferring the spatial relations, on the other hand, is useful for symbolic modeling of a world for high–level planning and question answering (e.g., an agent is asked to describe the spatial relation between two objects).

In cognitive science, there are two main approaches towards grounding spatial relations: the spatial template theory [176] and the constrained connectionist approach [177]. The spatial template theory assumes that humans utilize mental templates centered on a relatum and aligned with a reference frame to score regions of acceptability associated with a given relation (i.e., *good*, *acceptable*, and *bad*). On the other hand, the connectionist approach uses computational models such as Attentional Vector–Sum (AVS) model [177] to predict acceptability scores of spatial template theory for different spatial relations. From a computational perspective, the spatial template theory favors the rule–based approach, whereas the connectionist approach is

data–driven. Most systems introduced in literature follow the spatial template theory. These systems engineer a set of crisp templates [5], [71], [73], [140] or fuzzy templates [174], [178]–[180] to ground and infer the spatial relations.

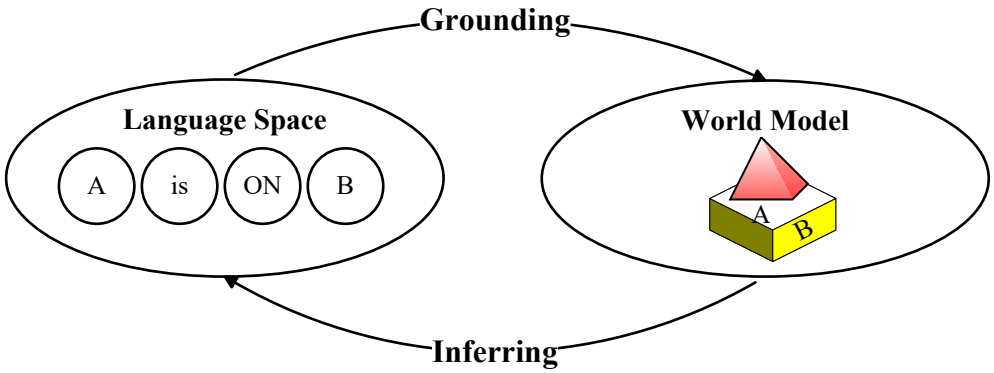

Figure 5.1. Coding and decoding the spatial relations between a spatial language in a symbolic space (language space) and a geometric representation in a visual space (world model)

These two models assume that spatial relations in language simply refer to a set of object positions in Euclidean space. However, it is shown that extra–geometric variables such as object knowledge are crucial in determining such relation [9]. Some works approach the mapping problem by trying to define an one–to–one mapping between language and world models using representations such as first–order logic. These models fail as there are many cases where the conditions for use of an arbitrary relation do not hold, but the use of that relation is still acceptable [9]. Another important aspect of many spatial relations is that they are view–centric and essentially depend on a position of a viewer (e.g., a camera in a virtual world).

Inspired by the cognitive model introduced in [9], we combine geometric features with semantic features of an object knowledge and follow the connectionist cognitive approach [177] to formulate both mapping tasks as supervised learning tasks. We compiled a dataset containing annotated geometric information of 3D spatial arrangements, object knowledge, and viewer information. We exploit this dataset to train two deep feed–forward neural networks (DNN). The first DNN is a regression model that addresses grounding task, whereas the second DNN is a

classifier that predicts a spatial preposition describing a given spatial arrangement. We also investigate the effect of object knowledge in both tasks.

Moreover, we construct a knowledgebase of co–occurrences of objects and spatial relations between them. This knowledgebase is represented as a Bayesian network with first order Markov assumption. We utilize it to suggest an object with its positions relative to a given object. This model facilitates world construction by auto–completing a user's design.

5.1 Background in Cognitive Science

Spatial relations have been extensively studied in cognitive science, neuroscience, and linguistics [126], [144], [181]–[187]. Cognitive scientists classify spatial relations into three general classes including basic relations, deictic relations, and intrinsic relations [176]. A basic relation takes one argument and expresses a position of an object with respect to a viewer. Space is represented in a crude binary manner and it is not necessary to identify, recognize or categorize the objects. Instead, an object is associated with a general token using a spatial index. Examples of basic spatial relations are "*something is there*" and "*it is not there*". In English, these relations exploit adverbs of place or intransitive prepositions and have implicit locatum in imperative forms. This relation cannot explicitly represent a relative relation between objects.

A deictic relation, on the other hand, takes more than one object as its arguments. In this relation, a position of a locatum is explicitly defined relative to one or more relatums with respect to a viewer's reference frame projected onto a relatum. Objects must be individualized to discriminate locatum from relatums. As an instance in "*The lamp is above the table"*, locatum (i.e., "*lamp"*) is explicitly discriminated from relatum (i.e., "*table"*). Finally, an intrinsic relation has two or more arguments and defines a position of a locatum with respect to a reference frame intrinsic to a relatum. This relation requires relatums with intrinsic reference frames that include "*top", "bottom"*, "*front"*, "*back"*, "*left"*, and "*right"* sides. Hence, it is necessary to either recognize or categorize objects to decide a relatum. For example, an object like a *ball* cannot be used as a relatum whereas objects such as "*human"*, "*house"*, and "*airplane"* can.

A reference frame is a central concept for positioning a locatum in all three classes. It is a 3D coordinate system that defines an origin, orientation (major axis), direction (up direction), and scale [176]. Representation of spatial reference frames can be classified into three classes

including allocentric, egocentric, and hybrid representations [188]. An allocentric representation defines a relation independent of a viewer. For example, "*The ball is 2 meters from the boy*" is valid regardless of where a viewer is standing or looking at. On the other hand, a viewer is the center of an egocentric representation in which a position of a locatum is encoded relative to her/him. For example, in "*The book is 3 meters to my right*", direction and distance are valid until the viewer does not move. Hybrid representation can be bifurcated into: (1) a representation that encodes a locatum position relative to a viewer while encoding the direction according to a relatum, and (2) a representation that encodes a locatum position relative to a relatum while encoding the direction according to a viewer's. "*The ball is 3 meters north of me*" and "*The ball is left of the book*" are examples of these two categories, respectively.

Furthermore, reference frames can be divided into three classes with respect to direction representation. These classes include absolute (environment–centered) reference frames, intrinsic (object–centered) reference frames, and relative (viewer–centered) reference frames [188]. In an absolute reference frame, a direction of a locatum relative to a relatum is expressed with respect to the earth (i.e., *north*, *south*, *east*, and *west*) or a screen window (e.g., *middle of the screen* and *top right corner of the window*). An intrinsic reference frame is defined based on the direction of an asymmetric relatum which embeds an intrinsic direction reference. This class is used in intrinsic spatial relations. Finally, a relative reference frame is aligned with a viewer's orientation and changes as soon as the viewer's orientation or direction is changed.

5.2 Approach

5.2.1 Inferring Spatial Relations

The inferring task is formulated as a multi–class classification task in which some discriminative features of two objects are fed to a classifier to decide a category of the spatial relation between them. Following the cognitive theory introduced in [9], input features consist of both geometric features (e.g., camera position and direction, object sizes, and object transformations) and semantic features (i.e., carrying object knowledge). Providing a learning model with these features lets it to implicitly learn the relations among viewer, relatum, locatum,

and the reference frame. Hence, rather than hand–crafting spatial templates, we let a learning model to automatically learn them.

We utilize a fully–connected feed–forward deep neural network (DNN) with a softmax layer to perform the classification task. Given the non–linearity of a DNN, it can encode various spatial templates within a few layers. The architecture of this model is shown in Figure 5.2. The output is a probability vector of 27 spatial relations and a predicted relation is one with the highest probability.

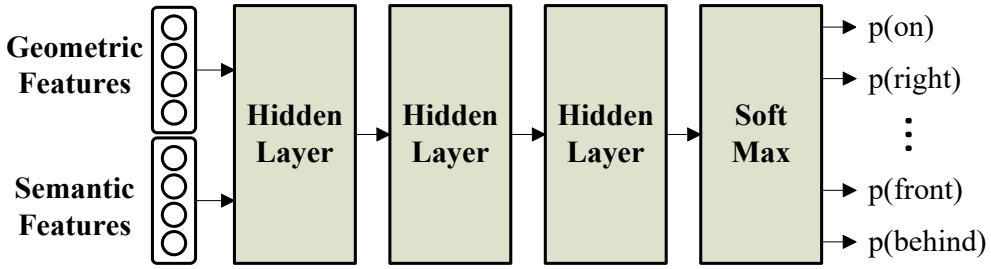

Figure 5.2. Proposed neural architecture for inferring spatial relations. Input is geometric and semantic features of two objects and output is a probability vector of candidate spatial relations

5.2.2 Grounding Spatial Relations

Grounding spatial relations, on the other hand, is formulated as a regression task. Given a viewer, a relatum, and a symbolic spatial relation, the goal is to predict a position of a locatum. Similarly, inspired by [9], we use both semantic and geometric features to train three fully–connected DNNs to predict a 3D position of a locatum (i.e., one DNN per axis). Moreover, to constrain the predictions, we feed each network with a prediction of its predecessor network. This helps a network to prune the search space by learning associations between position vectors (Pearson correlation coefficients for X–Y and Y–Z are 0.196 and 0.157, respectively). The architecture of the proposed neural model is shown in Figure 5.3.

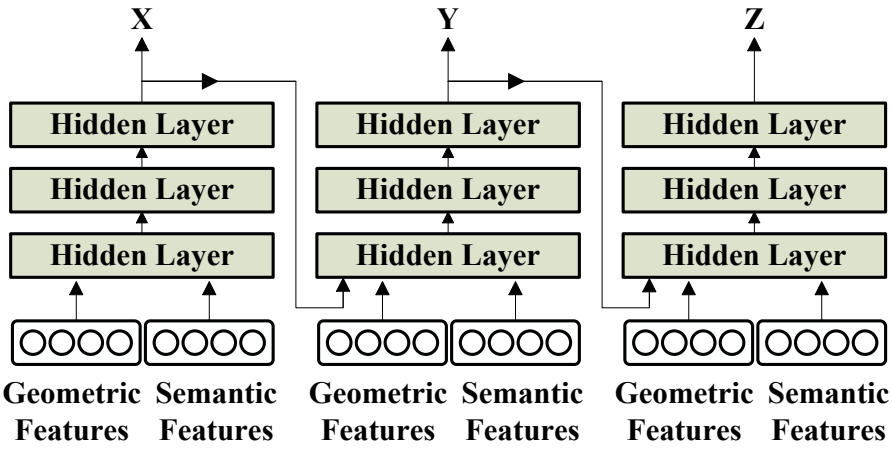

Figure 5.3. Proposed neural architecture for grounding spatial relations. Input is features of a located and a reference objects and output is a 3D position of the located object. Each network is fed with prediction of a predecessor network to propagate position constraints

5.2.3 Co–Occurrence Model

Co–occurrence knowledge is modeled as a Bayesian network with first order Markov assumption (i.e., a node is conditioned only on its direct neighbor objects). A node of this directed graph (v_o) represents an object, and an arc between two nodes $e_{o_1 \to o_2} :< r_i^{o_1:o_2}, n_{r_i}^{o_1:o_2} > \forall r_i \in \mathbf{R}$ represents a relation between two objects and number of occurrences of that relation between two objects. An example of this representation is shown in Figure 5.4.

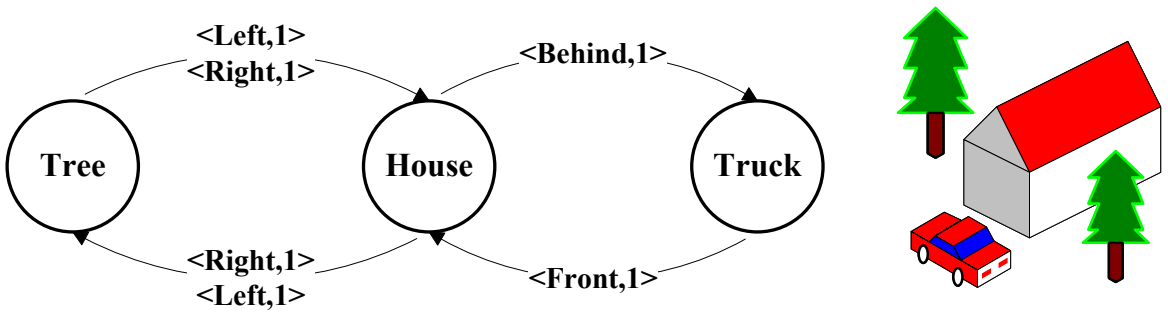

Figure 5.4. An example of co–occurrence model representation. There is one occurrence of *"front"* spatial relation between a truck as locatum and a house as relatum. Also, there is one occurrence of "*behind*" spatial relation between a truck as relatum and a house as a locatum. This distinction is important for proper recommendations

The co–occurrence model is constructed using a dataset of annotated 3D scenes discussed in section 5.5. In new scenarios, features of each pair of models within a scene are passed to the DNN model designed for inferring spatial relations and a symbolic spatial relation between those two models is extracted. Also, a reverse spatial relation is extracted from a small hand–crafted rule–base (e.g., *Front*(*A,B*)↔*Behind*(*B,A*)). Models are then looked up within the graph nodes and in case either of them is missing from the network, a new node is added to it. If a recognized relation between two objects has previously occurred, their corresponding count is incremented on an arc connecting the two objects. Otherwise, the relation is added to an arc and its count is set to 1. An important aspect of this network is that its construction is incremental and hence it can be expanded by new objects and relations being extracted while user designs new scenarios.

The purpose of co–occurrence model is to serve as a recommender system to facilitate a user's design scenarios by recommending objects and locating them within a world given a scenario context. To simplify sampling, we use the Markov assumption as follows: an object and its spatial relation are recommended only based on the last object that user added to the world (query object). In this setting, a recommended object is locatum whereas an added object is considered as relatum. Given this assumption, the recommending process can be formulated as follows.

$$\arg\max_{O_i, r_j} p(o, o_i, r_j) = p(o) \times p(o_i \mid o) \times p(r_j \mid o_i, o) \quad \forall o_i \in Neighbors(o),\ r_j \in R \tag{5.1}$$

O denotes the last object added to the world (query object), $o_i \in neighbors(o)$ is a member of a set of all objects that are directly connected to o, and $r_j \in R$ is a member of all defined spatial relations. Equation (5.1) is basically stating that given an object o, we want to select the most likely neighbor of o and the most likely spatial relation between them.

Given the Markov assumption, we already know that object o is selected and hence $p(o)=1$. $p(o_i|o)$ is occurrence probability of object o_i given that object o has already occurred. This probability is computed as follows.

$$p(o_i \mid o) = \frac{\sum_{r_k \in e_{O_i \to O}} n_{r_k}^{O_i \to O}}{\sum_{o_j \in N(o)} \sum_{r_k \in e_{o_j \to o}} n_{r_k}^{o_j \to O}} \tag{5.2}$$

$e_{oi\to o}$ denotes an arc connecting o_i to o, and $N(o)$ is a set of direct neighbors of o. $p(r_j|o_i,o)$ is occurrence probability of a spatial relation r_j between objects o_i and o computed as follows.

$$p(r_j \mid o_i, o) = \begin{cases} \dfrac{n_{r_j}^{O_i \to O}}{\sum_{r_k \in e_{O_i \to O}} n_{r_k}^{O_i \to O}}, & r_j \in e_{O_i \to O} \\ 0, & r_j \notin e_{O_i \to O} \end{cases} \tag{5.3}$$

Using (5.1) – (5.3), we compute all possible joint probabilities $p(r_j,o_i,o)$. Now, the question is how to select an object to be recommended. A simple approach is to recommend an object and relation that maximizes this joint probability (i.e., most probable object and spatial relation for an object). Nevertheless, it results in a constant recommendation for a specific object which is not plausible (e.g., If a user adds three desks to the world, all three recommendations will be putting a laptop on them). To address this issue, we borrow a stochastic selection strategy referred as ranking selection [189] from evolutionary computations. In this technique, a subset of all possible joint probabilities is randomly sampled and then an object–relation pair that maximizes the probability in that sampled sub–space is recommended to a user. This technique favors more likely pairs but gives chance to other pairs as well which in turn results in more diverse recommendations. It is noteworthy that recommending most probable object–relation pair is a specific case of ranking selection where sub–sample size equals the size of object–relation set.

5.3 Features

Inspired by [9], we develop two feature sets: geometric features and semantic features. Geometric features are designed in a way that they can address the view–centric nature of spatial relations, whereas semantic features can address the identity of an object (i.e., object knowledge). A geometric feature is a concatenation of 6 different features as follows.

1. Camera position:

$$\mathbf{\Phi_1} = \left[P_{cam}^x, P_{cam}^y, P_{cam}^z\right] \tag{5.4}$$

2. Camera direction:

$$\mathbf{\Phi}_2 = \left[D_{cam}^x, D_{cam}^y, D_{cam}^z\right] \tag{5.5}$$

3. Locatum model size:

$$\mathbf{\Phi}_3 = \left[S_{loc}^x, S_{loc}^y, S_{loc}^z\right] \tag{5.6}$$

4. Locatum transformation:

$$\mathbf{\Phi}_4 = \left[T_{loc}^{1,1}, T_{loc}^{1,2}, T_{loc}^{1,3}, T_{loc}^{1,4}, T_{loc}^{2,1}, T_{loc}^{2,2}, T_{loc}^{2,3}, T_{loc}^{2,4}, T_{loc}^{3,1}, T_{loc}^{3,2}, T_{loc}^{3,3}, T_{loc}^{3,4}\right] \tag{5.7}$$

5. Relatum model size:

$$\mathbf{\Phi}_5 = \left[S_{rel}^x, S_{rel}^y, S_{rel}^z\right] \tag{5.8}$$

6. Relatum transformation:

$$\mathbf{\Phi}_6 = \left[T_{rel}^{1,1}, T_{rel}^{1,2}, T_{rel}^{1,3}, T_{rel}^{1,4}, T_{rel}^{2,1}, T_{rel}^{2,2}, T_{rel}^{2,3}, T_{rel}^{2,4}, T_{rel}^{3,1}, T_{rel}^{3,2}, T_{rel}^{3,3}, T_{rel}^{3,4}\right] \tag{5.9}$$

The transformation of a given object is extracted from the top three rows of its transformation matrix in homogeneous coordinates as follows.

$$T = \begin{bmatrix} T_{1,1} & T_{1,2} & T_{1,3} & T_{1,4} \\ T_{2,1} & T_{2,2} & T_{2,3} & T_{2,4} \\ T_{3,1} & T_{3,2} & T_{3,3} & T_{3,4} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

In which $[T_{1,4},\ T_{2,4},\ T_{3,4}]^{\mathrm{T}}$ represents a translation of an object in x, y, and z directions, respectively, and $[T_{1,1}, T_{1,2}, T_{1,3}, T_{2,1}, \ldots, T_{3,3}]$ represents combined rotations and scaling of an object.

A semantic feature is defined as a concatenation of continuous word vectors of an object's annotations. Considering that word embeddings reflect similarities and dissimilarities between the words, they can be used to decide the semantic similarity between objects. For this purpose, we use pre–trained word vectors of GloVe and Word2Vec models discussed in section 2.3.

$$\mathbf{\Phi}_7 = \left[H_{GloVe}(w_1), H_{GloVe}(w_2), H_{GloVe}(w_3), H_{GloVe}(w_4), H_{GloVe}(w_5)\right] \tag{5.11}$$

$$\mathbf{\Phi}_8 = \left[H_{W2V}(w_1), H_{W2V}(w_2), H_{W2V}(w_3), H_{W2V}(w_4), H_{W2V}(w_5)\right] \tag{5.12}$$

$H_{GloVe}(w)$ and $H_{W2V}(w)$ are hash mappings between word–vector pairs of GloVe and Word2Vec models, respectively. It is assumed that an object annotation consists of maximum five words. Object annotations with fewer words are right–padded by *<PAD>* symbol and annotations with more than five words are trimmed from the left. This is because head words of a phrase are usually positioned to the right of a phrase (e.g., *the big* [*red cake with delicious blueberries*]).

Using geometric and semantic features, we define three feature sets for inferring task as follows.

$$\mathbf{F_1} = \left[\mathbf{\Phi}_1\mathbf{\Phi}_2\mathbf{\Phi}_3\mathbf{\Phi}_4\mathbf{\Phi}_5\mathbf{\Phi}_6\mathbf{\Phi}_7\right]^T \tag{5.13}$$

$$\mathbf{F_2} = \left[\mathbf{\Phi}_1\mathbf{\Phi}_2\mathbf{\Phi}_3\mathbf{\Phi}_4\mathbf{\Phi}_5\mathbf{\Phi}_6\mathbf{\Phi}_8\right]^T \tag{5.14}$$

$$\mathbf{F_3} = \left[\mathbf{\Phi}_1\mathbf{\Phi}_2\mathbf{\Phi}_3\mathbf{\Phi}_4\mathbf{\Phi}_5\mathbf{\Phi}_6\right]^T \tag{5.15}$$

Features $\mathbf{F_1}$ and $\mathbf{F_2}$ utilize both geometric and semantic features. The only difference between them is the utilized word vector model as the semantic feature. Feature $\mathbf{F_3}$ on the other hand only uses geometric features. Comparing these features can demonstrate the correlation between a spatial relation and the object's knowledge.

The same features except $\mathbf{\Phi_4}$ are used for grounding task. The goal is to predict a position of a located object $\left[x_{loc}, y_{loc}, z_{loc}\right] = \left[T_{loc}^{1,4}, T_{loc}^{2,4}, T_{loc}^{3,4}\right]$. Note that it is assumed that the scale of a located object is available (please refer to chapters 6 and 7). Therefore, the following features are utilized for this task.

$$\mathbf{F_4} = \left[\mathbf{\Phi}_1\mathbf{\Phi}_2\mathbf{\Phi}_3\mathbf{\Phi}_5\mathbf{\Phi}_6\mathbf{\Phi}_7\right]^T \tag{5.16}$$

$$\mathbf{F_5} = \left[\mathbf{\Phi}_1\mathbf{\Phi}_2\mathbf{\Phi}_3\mathbf{\Phi}_5\mathbf{\Phi}_6\mathbf{\Phi}_8\right]^T \tag{5.17}$$

$$\mathbf{F_6} = \left[\mathbf{\Phi}_1\mathbf{\Phi}_2\mathbf{\Phi}_3\mathbf{\Phi}_5\mathbf{\Phi}_6\right]^T \tag{5.18}$$

5.4 Dataset

The scene dataset was collected by recruiting annotators on Amazon Mechanical Turk (AMT). We presented an annotator with a sample scene of a 3D environment and asked her/him

to describe spatial relations within a scene in form of locative expressions [60]. For each task, an annotator is provided with a snapshot of a 3D scene and a set of object names within a scene and is asked to provide at least 20 sentences consisting of at least 5 distinct spatial prepositions. S/he is also asked to annotate relatums, locatums, and spatial pivots. Annotators are provided with 196 snapshots of 132 scenes. These scenes are collected from the Stanford Scene Database [74]. There are 1,741 3D models used in these scenes where on average each scene consists of 27 models with a standard deviation of 18.

Also, to count the effect of camera position and direction, we augment the dataset by taking snapshots of the same scene by setting the camera to different positions and directions. An example of such augmentation is shown in Figure 5.5. The collected dataset contains 3,920 sentences consisting of 37,836 terms and 987 words. Annotators described spatial relations using 27 distinct spatial pivots. Occurrence counts of these pivots are shown in Figure 5.6. As shown, only 5 of them have occurred more than 200 times.

Figure 5.5. Four snapshots from the same scene presented to annotators on Amazon Mechanical Turk. Given the view–centric nature of spatial relations, we augment the annotations by taking snapshots of the same scene with different camera positions and directions

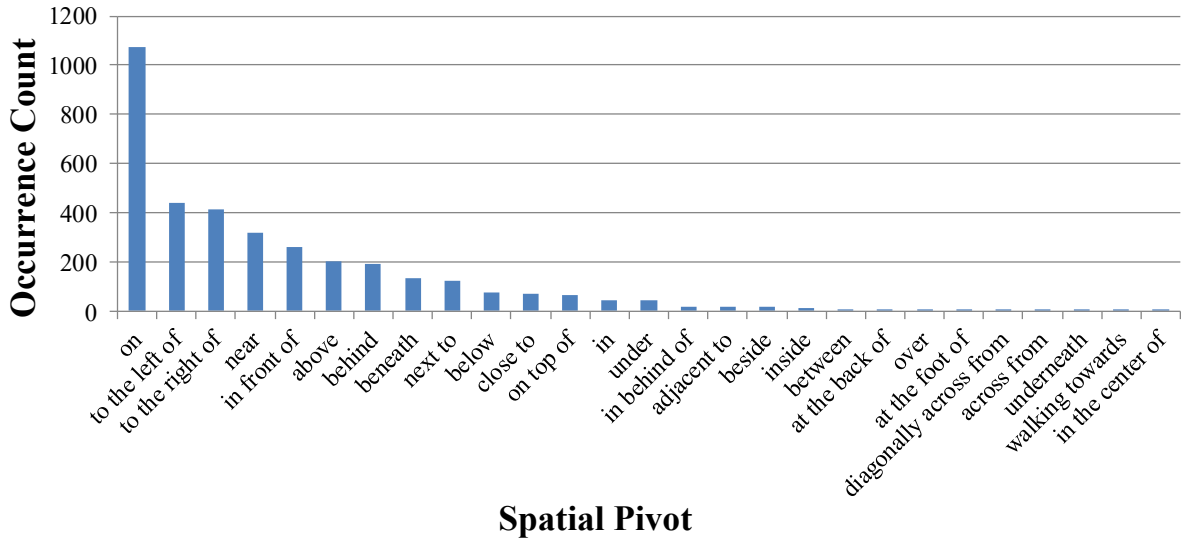

Figure 5.6. Distribution of spatial pivot occurrences in the scene dataset. Some pivots appear more frequently than others. Only 5 of them have more than 200 occurrences

For grounding task, we train a model per spatial pivot. Hence, we split the dataset based on spatial pivots (i.e., one dataset for *on*, one dataset for *to the left of*, etc.). We further split the datasets by 70%–30% as train and test sets, respectively. On the other hand, we train a single model for inferring task. Similarly, we split the dataset by 70%–30% using stratified sampling (i.e., classes in both train and test sets have roughly same ratios).

5.5 Experimental Results

5.5.1 Inferring Task

Data is preprocessed as follows. First, data in each dimension is normalized by subtracting data points from the mean and dividing them by the standard deviation of that dimension. For universal word embeddings, missing words in pre–trained models are randomly generated by sampling each dimension from U~[–1,+1]. The dimensionality of pre–trained vectors is also reduced from 300 to 10, 50, 100, and 200 using Principal Component Analysis (PCA) to investigate the effect of different dimensions. To investigate the performance of other models and compare them with proposed DNN model, we utilize the random forest classifier,

bagging classifier, Ada–boost classifier, RBF–SVM, K–NN, and logistic regression models as well. Hyper–parameters of these models are tuned using a random search strategy (Appendix D).

The DNN model is trained using Adam stochastic optimizer [169] with a learning rate of 3E−4 over the mini–batches of size 250 and 50,000 epochs. Mini–batches are uniformly sampled with replacement from the training set. We utilize cross–entropy loss function with one–hot output representation, dropout regularization [170] with a probability of p=0.5, and batch normalization [171] on input layer. We also utilize Rectified Linear Units (ReLU) in hidden layers and a soft–max function in the output layer. DNN model is defined as a four–layer network with 3 fully–connected hidden layers (800+1200+300 neurons).

We first investigate the effect of a dimension of word vectors on predictions. This effect on a bagging classifier trained on the "*on*" preposition is shown in Figure 5.7. Both word vector models achieve the best performance when the dimension is set to 50. This is associated with the size of a dataset and hence is subject to change. Also, both models achieve a similar F_1–score using this setup. We observe the same behavior for other models as well.

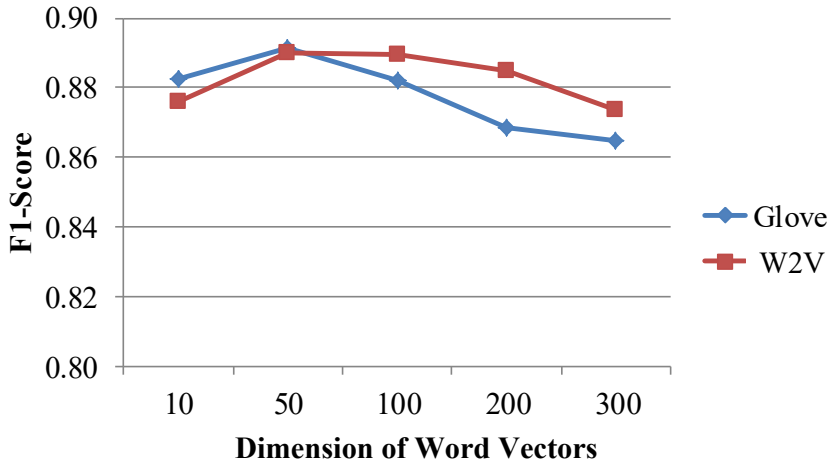

Figure 5.7. Effect of the dimension of word vectors on F_1-score of a bagging classifier trained for inferring spatial relations. Both GloVe and Word2Vec models result in highest score when their size is set to 50

We also investigate the effect of proposed semantic feature on inferring task. For this purpose, we compare F_1–scores achieved by training models on features $\mathbf{F_1}$ (semantic + geometric) and $\mathbf{F_3}$ (geometric) for "*on*" preposition. Results are shown in Figure 5.8.

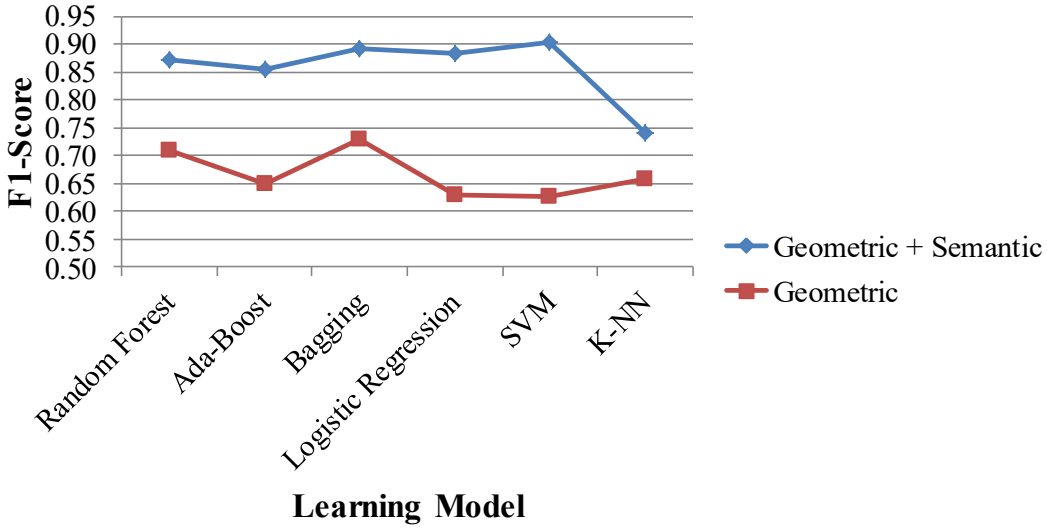

Figure 5.8. Effect of proposed semantic feature on inferring task. Results suggest that embedding this feature with geometric features significantly enhances the performance

As shown, exploiting object knowledge significantly improves the F_1–score in all models (e.g., in the RBF–SVM model, the F_1–score is improved by an outstanding enhancement of 43.75%). These results support the cognitive theory of spatial relations proposed in [9]. Fine–grained results for preposition "*on*" and coarse–grained results (i.e., all 27 classes) achieved by learning models using the $\mathbf{F_1}$ feature set (semantic + geometric) are shown in Tables 5.1 and 5.2, respectively. Also, fine–grained results of five most frequent prepositions in the dataset learned by DNN model are shown in Table 5.3. These results suggest that the DNN model outperforms other models in terms of precession, recall, and F_1–score measures. Furthermore, the fine–grained results suggest that at least 1000 samples are required for a spatial pivot to achieve an F_1–score over 0.90 (i.e., a good performance for practical applications).

Table 5.1. Fine–grained classification results for preposition "*on*" using proposed feature set. Proposed DNN model trained on our proposed feature outperforms other models in terms of precision, recall, and F_1-score

Classifier	Precision	Recall	F_1–score
Random Forest	0.8127	0.9400	0.8717
Ada–Boost	0.8323	0.8767	0.8539
Bagging	0.8358	0.9333	0.8819
Logistic regression	0.8383	0.9333	0.8833
RBF–SVM	0.8606	0.9467	0.9016
K–NN	0.5988	0.9700	0.7405
Proposed DNN	**0.8823**	**0.9763**	**0.9269**

Table 5.2. Coarse–grained classification results using proposed feature on 27 spatial pivots. Our proposed DNN model trained on proposed feature outperforms other models in terms of precision, recall, and F_1-score.

Classifier	Precision	Recall	F_1–score
Random Forest	0.6041	0.6188	0.6024
Ada–Boost	0.5181	0.5275	0.5193
Bagging	0.6168	0.6255	0.6145
Logistic regression	0.5614	0.5812	0.5690
RBF–SVM	0.5667	0.5839	0.5721
K–NN	0.5367	0.5597	0.5113
Proposed DNN	**0.6285**	**0.6412**	**0.6348**

Table 5.3. Fine–grained classification results of proposed DNN model trained on semantic and geometric features for top 5 most frequent spatial pivots. This suggests that at least 1000 samples are required for a spatial pivot to achieve an F_1–score over 0.90.

Classifier	#Occurrences	Precision	Recall	F_1–score
On	1074	**0.8823**	**0.9763**	**0.9269**
To the left of	440	0.4133	0.4429	0.4276
To the right of	416	0.4194	0.4333	0.4262
Near	317	0.4944	0.4632	0.4783
In front of	262	0.5957	0.3111	0.4088

Moreover, we test the statistical significance of the results using McNemar's test. The results shown in Table 5.4 suggest that the classification results are statistically significant.

Table 5.4. Results of the McNemar's statistical test on the inferring task

Classifier Pairs	χ^2 Estimation	P
DNN ⊗ Random Forest	28.928	<0.005
DNN ⊗ Ada–Boost	22.755	<0.005
DNN ⊗ Bagging	30.947	<0.005
DNN ⊗ Logistic regression	16.488	<0.005
DNN ⊗ RBF–SVM	14.886	<0.005
DNN ⊗ K–NN	129.188	<0.005

5.5.2 Grounding Task

Considering that the grounding task is a regression problem, we evaluate the trained models using two regression measures including Mean Square Error (MSE) and R^2–score (i.e., the coefficient of determination). MSE is defined as follows.

$$MSE = \frac{\sum_{i=1}^{N}\left((x_i^d - x_i^p)^2 + (y_i^d - y_i^p)^2 + (z_i^d - z_i^p)^2\right)}{N} \tag{5.19}$$

where N is the number of samples, x^d is a desired value of x, and x^p is a predicted value of x.

R^2–score is computed as follows. $\bar{x}$ denotes the mean value of observed x.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left((x_i^d - x_i^p)^2 + (y_i^d - y_i^p)^2 + (z_i^d - z_i^p)^2\right)}{\sum_{i=1}^{N}\left((x_i^d - \bar{x})^2 + (y_i^d - \bar{y})^2 + (z_i^d - \bar{z})^2\right)} \tag{5.20}$$

We use eleven regression models including random forest, Ada–boost, bagging, linear, Automatic Relevance Determination (ARD), ElasticNet, Lasso, Least–Angle Regression (LARS), Support Vector Regression (SVR), K–NN, and DNN (MLP) regression models to investigate the performance. All models except DNN are implemented using scikit–learn toolkit [167] and DNN model is implemented using TensorFlow [168]. The hyper–parameters of these models are tuned using random search strategy (Appendix E).

The DNN model developed for grounding task has the same learning rate, mini–batch size, number of epochs, regularization, and activation function as the DNN model developed for

inferring task. However, we use a mean square loss function with continuous output representation (i.e., each DNN has only one output). The model is defined as a four layer fully–connected network (1000+500+200 hidden neurons + 1 output neuron).

We first investigate the effect of the dimension of word vectors on regression performance. Results for SVR model are shown in Figure 5.9. As shown, the best performance for both word vector models in regression task (i.e., minimum MSE) is achieved with the minimum vector size (i.e., vector size of 10). Also, as shown Word2Vec model slightly outperforms GloVe model in this task.

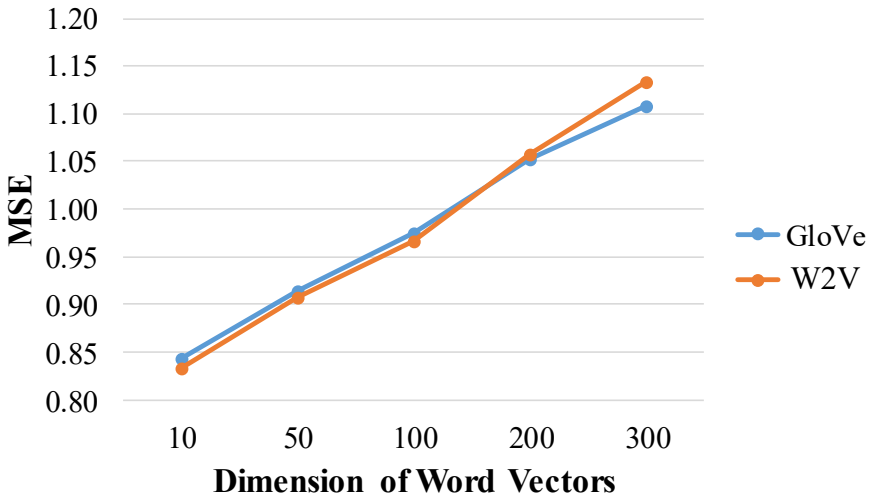

Figure 5.9. Effect of the dimension of word vectors on MSE measure of a support vector regression model trained for spatial grounding task. Word2Vec model slightly outperforms GloVe model

Regression results using geometric features, and a combination of geometric and semantic features is shown in Figure 5.10. As illustrated, once more results support the cognitive theory proposed in [9] and combining semantic features with geometric features enhances the performance. Finally, results of various regression models along with our proposed DNN model are shown in Table 5.5. As shown, the DNN model outperforms other models in terms of MSE and R^2–score measures.

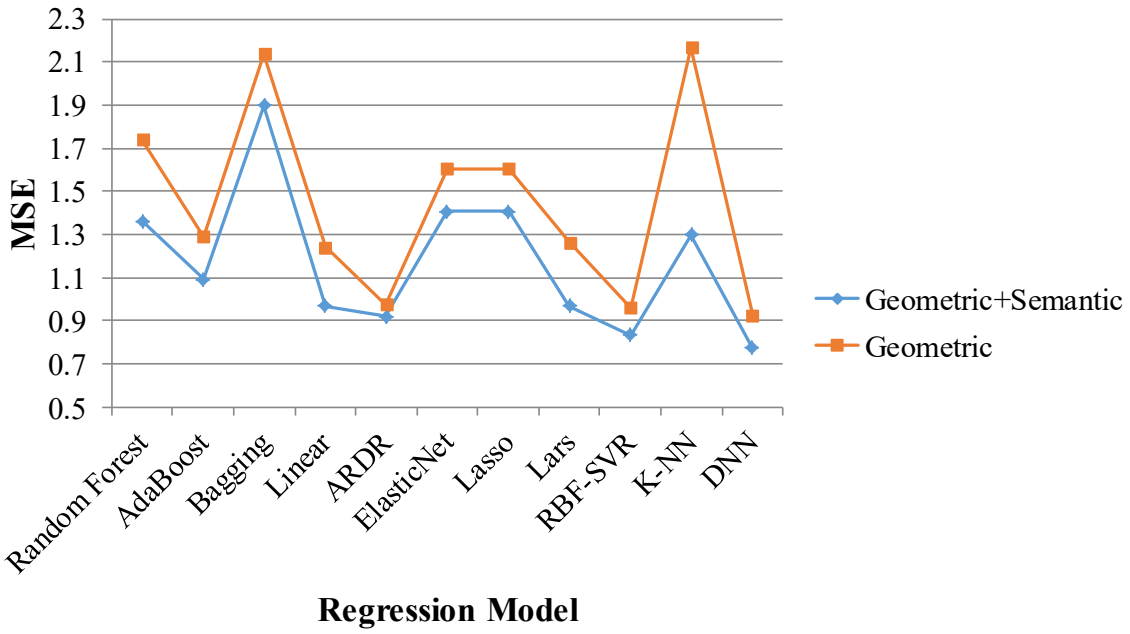

Figure 5.10. Effect of adding semantic features on grounding task tested with 11 regression models. All models illustrate better generalization having lower Mean Square Error (MSE) on test set when they are trained with both semantic and geometric features

Table 5.5. Results of various regression models on spatial grounding task using proposed combination of semantic and geometric features. Our proposed DNN model outperforms other models in terms of MSE and R^2–score measures

Regression Model	MSE	R^2–score
Random Forest	1.3624	0.3147
Ada–Boost	1.0911	0.3588
Bagging	1.8969	0.3621
Linear	0.9678	0.3413
ARD	0.9197	0.3741
ElasticNet	1.4046	–0.0240
Lasso	1.4046	–0.0240
Lars	0.9678	0.3413
RBF–SVR	0.8324	0.4286
K–NN	1.2972	0.1171
DNN	**0.7723**	**0.4513**

5.5.3 Recommendation Task

We constructed this model in an unsupervised manner on the collected scene dataset. The resulted model consists of 697 objects (e.g., *desk*, *plate*, *shampoo*, *dog*, etc.). The statistics of this model are shown in Table 5.6. An arbitrary relatum object in this model has an average of 4.85 candidate pairs to be recommended with a standard deviation of 8.56. Also, "*desk*" with 131 candidate object–relation pairs has the highest variety. Among its candidates <*keyboard, on*> pair with a probability of 0.046 is the most probable pair.

Table 5.6. Statistics of the recommender model. Object-Relation pair denotes the number of candidate pairs of a relatum object

	Object–Relation	Object	Relation
Mean	4.85	4.36	2.61
SD	8.56	7.40	1.96
Min	1	1	1
Max	131	104	13

Recommendations are sensitive to the size of a sampled subset selected for ranking selection. A small subset results in more diversity with higher randomness whereas a big subset follows an underlying distribution with less diversity. To address this trade–off, we set a subset size to a quarter of population size. The top five candidates recommended by ranking selection for the *kitchen countertop*, *desk*, *round table*, and *widescreen TV* are shown in Figures 5.11–5.14, respectively (we repeated sampling for 100 times).

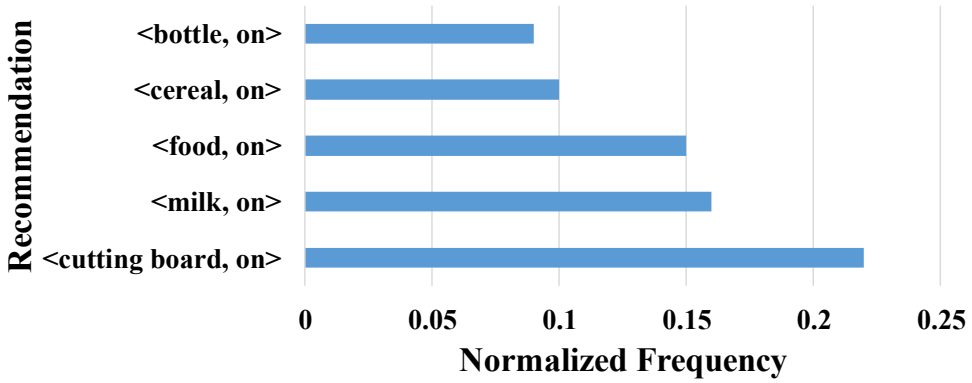

Figure 5.11. Top five recommendations for "*kitchen countertop*"

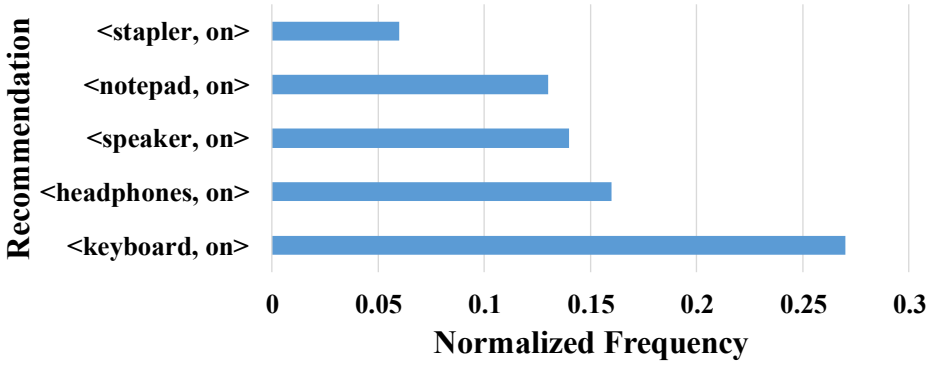

Figure 5.12. Top five recommendations for "*desk*"

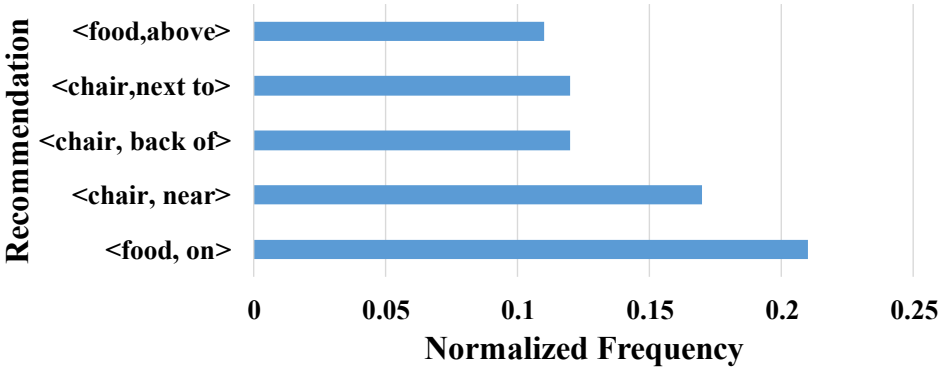

Figure 5.13. Top five recommendations for "*round table*"

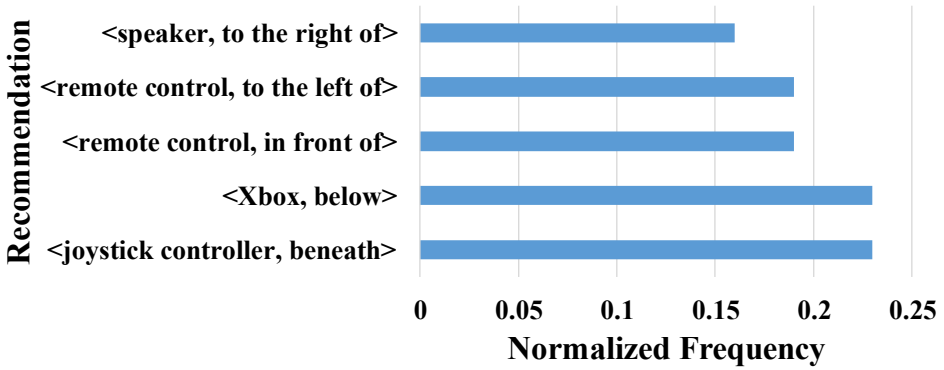

Figure 5.14. Top five recommendations for "*widescreen TV*"

The recommendation task requires subjective evaluations. We extracted the top 5 candidate pairs for 50 objects and asked four users to score how likely the recommended object–

relation pairs are based on the Likert scale (i.e., 1: not likely, 2: less likely, 3: more or less likely, 4: likely, and 5: very likely). The users ranked 93.3% of the recommendations as very likely and 6.6% of recommendations as more or less likely. Those objects that are ranked as less likely are objects that rarely co–occur with a query object. As an example, a collection of swords occurring behind a dining table is a rare event, which was one of the recommendations. It only appeared in one scene and due to probabilistic nature of recommendations, it was recommended to users. Although such co–occurrences are less likely, yet recommending them improves the naturalness of the recommendations.

We addressed spatial relations in visual and symbolic spaces and mapping between these two spaces using deep feed forward models. Moreover, inspired by a theory from cognitive science, we developed a set of semantic and geometric features. Experimental results suggest that using these features along with deep neural models outperforms other models in both tasks. We also developed a co–occurrence model and a recommender engine using ranking selections. Results suggest that the recommendations made by this model are plausible.

Chapter 6. Offline–Learning Physical Attributes

In the previous two chapters, we thoroughly addressed various processes on spatial relations in both symbolic and visual spaces. Another important category of common–sense knowledge is represented by the physical attributes of objects. These attributes such as static attributes (e.g., *size* and *weight*) and dynamic attributes (e.g., *speed*) are essential for 3D world modeling. Size information is required to render plausible relative sizes of 3D models (e.g., *a cat is smaller than a table*); weight information is required by physics engine to generate realistic dynamics; and speed distribution over objects is required to generate plausible relative motions (e.g., a *turtle moves slower than a cat*). Yet this knowledge is missing from available common–sense knowledgebases and hence they cannot answer simple questions such as: "*is a microwave oven bigger than a spoon?*" or "*is a feather heavier than a king size mattress?*". Considering the huge number of object categories and varieties in each category, it is not practical to populate a knowledgebase with hand–crafted facts about physical attributes. To bridge this gap, we exploit web data from heterogeneous resources to train a set of models to estimate the physical attributes.

3D models are the building blocks of a virtual world and huge amounts of them are available on web repositories such as 3D Warehouse, Archive3D, TurboSquid, and Artist3D. By browsing these online models, we observe the followings: (1) for the most common objects in real–life, corresponding 3D models are available; (2) models are created in standard formats, (3) they have a consistent upright orientation, (4) they have natural dimension ratios (e.g., height to width ratio), (5) their sizes are inconsistent with their corresponding real–life objects (i.e., absolute sizes), (6) they mostly carry some annotations, (7) they do not carry information regarding physical attribute (e.g., *weight* or *speed*).

Information regarding physical attributes of real–life objects, on the other hand, is not easily accessible and if any, available information is not in a canonical form (i.e., different units and measurements, constant and distributional representations, etc.). Hence, extracting and assigning them to corresponding 3D models is a challenging task. We exploit web content mining with several criteria to address this challenge. First, it should discriminate physical

attributes of a given object. Second, it should extract numeric values corresponding to physical attributes from the web content while minimizing noise. Third, it should model the extracted values as a distributional representation to support sampling.

We propose two approaches. The first approach is offline–learning. It is assumed that there are repositories on the web (e.g., Walmart website) containing data about some physical attributes of objects. Using this data, it learns distributions of physical attributes using Gaussian Mixture Models (GMM). We also train GMMs on a 3D model database to model size ratios in order to reduce the noise of GMMs trained on the web data. Moreover, we propose a Monte Carlo inference mechanism to infer a set of comparative relations among the physical attributes.

The second approach, online–learning, is designed to cover the missing knowledge that cannot be retrieved using the offline–learning. This approach trains a classifier to predict physical attributes of a given object and then trains a deep Convolutional Neural Network (CNN) to extract the values of physical attributes of unseen objects from web search results. We discuss the offline–learning in this chapter and explain the online–learning in next chapter.

6.1 Related Work

Cyc [158] is an integrated knowledgebase and inference engine with one million hand–crafted assertions and axioms of common–sense knowledge formulated in CycL language. ConceptNet [21] is an automatically generated knowledgebase that contains spatial, physical, social, temporal, and psychological aspects of common–sense knowledge and consists of 1.6 million assertions represented as a semantic network. WordNet [19] is a lexical resource with 101,863 nouns that provides the common–sense knowledge as a lexical taxonomy using hypernyms and hyponyms (i.e., *X is a kind of Y*) and holonyms and meronyms (i.e., *X is a part of Y*) relations within in a Directed Acyclic Graph (DAG) structure. Nevertheless, none of these knowledge–bases provide information regarding physical attributes of object categories.

ShapeNet [190], [191] is an ongoing project for creating an annotated repository of 3D models. It currently contains 3,000,000 models out of which 220,000 models are classified using the WordNet taxonomy. The goal of this project is to annotate models by language–related annotations, geometric annotations, functional annotations, and physical annotations. Annotating is carried out algorithmically and then verified using crowd–sourcing. Absolute sizes are

estimated using the approach proposed in [75] and weights are estimated by multiplying an object's volumes by its material densities.

In [75], univariate normal distributions are used to estimate bounding box diagonals of 3D model categories (i.e., uniform scaling factor). It uses prior absolute sizes of indoor objects and relative sizes of corresponding objects within 3D scenes to learn the distribution parameters. Taxonomy merging is performed by computing cosine similarity between representative words of a model and candidate synsets using Term Frequency–Inverse Document Frequency (TF–IDF) measure. In [192], relative sizes of objects are estimated by maximizing the joint likelihood of textual and visual observations. This method utilizes a graph representation with nodes corresponding to univariate log–normal distributions of object sizes and arcs corresponding to relative sizes between objects that frequently co–occur. Textual information is extracted using search query templates and visual information is extracted by detecting objects and estimating depth adjusted ratio of areas of their bounding boxes. These works only focus on object scales and try to extract a uniform scaling factor. Hence, they cannot address dimensions individually (e.g., *wider*, *longer*, *taller*).

In [193], WordNet and syntactic analysis are used to extract comparative common–sense knowledge (e.g., *an apple is smaller than a pineapple*) from a big web corpus. Extracted knowledge is disambiguated using Integer Linear Programming (ILP) and semantic coherence score, and then represented as a set of triplets (e.g., <*apple, smaller, pineapple*>). Although this approach is useful for question–answering tasks, it is not applicable to computer graphics or robotics tasks as it does not provide numeric information.

6.2 Offline–Learning

In offline–learning, it is assumed that collections of semi–structured data of interest are available on the web. The goal is to locate and parse the data, align it to a reference taxonomy and then model its distribution. The schematic of offline–learning is illustrated in Figure 6.1. As shown, it is a pipeline of three components including a semantic merging unit, a logistic regression model, and a GMM. Semantic merging aligns the harvested heterogeneous categories of objects with WordNet taxonomy using a semantic distance measure in an unsupervised manner. The logistic regression model is trained to augment the hypernym hierarchy of WordNet

by inserting new categories that are missing from it. Finally, GMM models are trained using Expectation Maximization (EM) to model the distributions of physical attributes of different object categories.

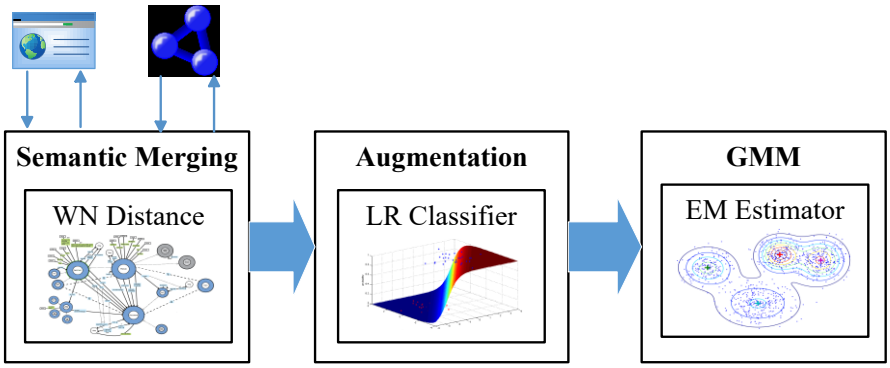

Figure 6.1. Schematic of offline–learning of physical attributes. Semantic merging uses a WordNet semantic distance to merge heterogeneous taxonomies into WordNet taxonomy, Augmentation is used to insert object categories that are missing from WordNet using a Logistic Regression (LR) Classifier, and GMMs are trained to represent distributions of object categories

6.3 Harvesting the Web Data

The first step towards the automatic discovery of common–sense knowledge is to collect data. We exploit data embedded within the web documents to harvest such data. We designed two mechanisms for this purpose: batch harvesting and online harvesting. Batch harvesting mechanism crawls a set of predefined websites and harvests data from crawled documents. Online harvesting, on the other hand, focuses on a specific object and uses the data retrieved from a search engine such as Google. In this chapter, we focus on the former approach.

We are interested in absolute sizes and weights of different object categories. To collect relevant data, we used the websites of Walmart, Costco, IKEA, and BestBuy retail stores. These websites contain data associated with a hierarchy of object categories along with instance objects, their sizes, weights, and sample images as well as other attributes. To systematically extract this data, we customized a crawler and a Document Object Model (DOM) parser based

on DOM template of each website. We crawled over 200K pages and extracted 108,105 items from them.

Considering the noisy nature of the web, we preprocessed the extracted items using some regular expressions. These regular expressions are used to extract numerical values of attributes, standardize units, and normalize dimensions. Dimensions are rearranged to *length*×*width*×*height* format from other formats (e.g., *area*×*height* and *width*×*depth*×*thickness*) and units are converted to the SI base units (i.e., *meter* for length and *gram* for weight). Furthermore, duplicate items and items with missing physical attributes are filtered out. The final dataset contains 69,433 objects classified into over 500 categories. Some information regarding this dataset is summarized in Table 6.1. Max depth denotes the maximum depth of a taxonomical hierarchy. We also collected a dataset of dimension ratios of 220,000 classified 3D models from ShapeNet and used it to post–process trained models on the web dataset.

Table 6.1. Summary of data harvesting and filtering results for offline–learning physical attributes of various object categories

	Walmart	**Costco**	**IKEA**	**BestBuy**	**Total**
#Raw Items	45,946	3,870	19,182	39,152	108,150
#Valid Items	42,732	3,534	4,232	18,935	69,433
#Categories	1,260	530	378	722	>500
Max Depth	4	4	4	6	———

6.4 Semantic Taxonomy Merging

Data harvested from the web comes with heterogeneous taxonomical hierarchies and granularities. As an example, taxonomical hierarchies of a *washing machine* in harvested data and WordNet are as follows. Note that hierarchy of BestBuy is more fine–grained than WordNet and Walmart.

- **WordNet**: [*entity*]→[*physical entity*]→[*object, physical object*]→[*whole, unit*]→[*artifact*]→[*commodity, trade good, good*]→[*consumer goods*]→[*durables, durable goods, consumer durables*]→[*appliance*]→[*home appliance*]→[*white goods*]→[*washer, automatic washer, washing machine*]
- **Walmart**: [*Appliances*]→[*Home Appliances*]→[*Washing Machines*]

- **BestBuy**:
  - [*Appliances*]→[*Washers, Dryers & Laundry Accessories*]→[*Washers*]→[*Front Load Washers*]
  - [*Appliances*]→[*Washers, Dryers & Laundry Accessories*]→[*Washers*]→[*Top Load Washers*]

To utilize heterogeneous information, one should first define a scheme to integrate hierarchies into a reference taxonomy. Taxonomy merging refers to a process of integrating two or more taxonomies on the same subject by eliminating duplicate terms and enhancing the taxonomy using terms from all taxonomies. To address this, we should determine: (1) a reference taxonomy; (2) a mechanism to enrich local taxonomies with representative terms and phrases; (3) a semantic similarity measure between taxonomies; and (4) a mechanism to extend the reference taxonomy. We exploit the WordNet's *IS–A* taxonomy (i.e., hypernym and hyponym relations) as our reference taxonomy. The latter three requirements are addressed as follows.

6.4.1 Enriching Local Taxonomies

Instances of each category contain relevant information about that category. For example, the hierarchical category of *air purifier* in BestBuy taxonomy ([*Home*]→[*Appliances*]→[*Air Conditioning & Heating*]→[*Air Purifiers*]→[*Air Purifiers*]) contains 50 instances such as: (*Honeywell AirGenius 5 Oscillating Air Cleaner and Odor Reducer with Permanent Filter*), (*Gern Guardian 3–in–1 Air Cleaning System with True HEPA Filter – Black Onyx*), and (*Germ Guardian Pluggable UV–C Air Sanitizer – Silver*). Extracting representative terms and phrases such as *air cleaning system*, *air sanitizer*, *odor reducer*, and *air cleaner* from instances, and combining them with category headers (e.g., *air purifiers*) can significantly enhance the accuracy of taxonomy merging.

For this purpose, a preprocessing step is performed to reduce noise by normalizing the text, lemmatizing plural nouns, and eliminating stop words. Unigrams, 1–skip–bigrams, and 2–skip–trigrams are then extracted from preprocessed instances and category names using the NLTK toolkit [163]. Synonym concepts of extracted skip–grams are also retrieved from ConceptNet and added to skip–grams. The next step is to select the top $k=10$ discriminative candidates denoted by Ψ_C. We only consider those candidates that are children of the *physical object* synset in WordNet. We used three feature selection methods including mutual information

(MI), χ^2–test, and normalized TF–IDF to select these candidates. The MI measure is defined as follows.

$$MI(C;T) = \sum_{c\in\{C,\bar{C}\}} \sum_{t\in\{T,\bar{T}\}} p(c,t)\log_2\left(\frac{p(c,t)}{p(t)\times p(c)}\right) \tag{6.1}$$

C denotes a category and T denotes a skip–gram. χ^2–test is defined as follows.

$$\chi^2(C;T) = \sum_{c\in\{C,\bar{C}\}} \sum_{t\in\{T,\bar{T}\}} \frac{\left(N(c,t)-E(c,t)\right)^2}{E(c,t)} \tag{6.2}$$

$N(c,t)$ denotes an observed frequency of t in category c and $E(c,t)$ denotes an expected frequency of t in category c. Finally, normalized TF–IDF is defined as follows.

$$Ntfidf(C,T) = \left(\gamma + (1-\gamma)\frac{tf(C,T)}{tf_{\max}(C)}\right)\times\log\left(1+\frac{|C|}{N_T}\right) \tag{6.3}$$

$tf(C,T)$ is frequency of term T in category C, $tf_{max}(C)$ is maximum frequency of a term in category C, $|C|$ is the number of classes (i.e., two in this case), N_T is the number of classes that T appears in, and γ is a smoothing term set to 0.4 as advised by [194].

6.4.2 Semantic Similarity

Given a query q, WordNet retrieves a triplet $H_i(q)=<s,g,h>$, $i=1...M$ where s_i denotes a synset, g_i denotes a gloss, h_i denotes a hypernym hierarchy, and M denotes the polysemy degree of query q. We feed the extracted skip–grams of each category ($q\in\Psi_C$) to WordNet and extract $H_i(q)$ for each skip–gram. This produces a set $T_C=\{H_1(q_1),\ldots,H_M(q_N)\}$, $q_j\in\Psi_C$. We then filter out synsets that have the *abstract entity* in their hypernym hierarchies from T_C. Assuming N skip–grams with average polysemy degree of M for a category, there are $N\times M$ candidates to consider. The goal is to find a h_i in WordNet which has the maximum semantic similarity with Ψ_C. We extract unigrams, 1–skip–bigrams, and 2–skip–trigrams of $H_i(q_j)$ and denote this set by Ψ_R. Given these assumptions, an unsupervised method is required to select a candidate that maximizes the likelihood of selecting a correct category as follows.

$$\Psi = \arg\max_{h_{ij}}\left(Sim(\Psi_R,\Psi_C)\right) \tag{6.4}$$

Ψ denotes a selected hierarchal category and $Sim(\Psi_R, \Psi_C)$ measures the semantic similarity between taxonomical hierarchies Ψ_R and Ψ_C. We define this measure as a normalized pairwise semantic similarity between representative skip–grams of a source category and candidate skip–grams of a reference category as follows.

$$Sim(\Psi_R, \Psi_C) = \frac{\sum_{t_r \in \Psi_R} \sum_{t_c \in \Psi_C} sim(t_r, t_c)}{|\Psi_R| \times |\Psi_C|} \tag{6.5}$$

$|\Psi_R|$ denotes the number of representative skip–grams of Ψ_R and $sim(t_r,t_c)$ is the semantic similarity between two skip–grams. To decide the similarity measure between two skip–grams, we examined seven lexicon–based semantic similarities and two continuous vector space semantic similarity measures. Lexicon similarity measures include six WordNet–based and one ConceptNet–based similarities [14], [195]. These semantic similarities are defined as follows.

(1) **Path similarity** assumes a shorter path between two senses implies their semantic similarity. It is computed by the number of edges in the shortest path connecting senses in a hypernym relation (i.e., $dis(C_1,C_2)$) as follows.

$$sim_{path}(C_1, C_2) = -\log\big(dis(C_1, C_2)\big) \tag{6.6}$$

(2) **Leacock–Chodorow similarity** normalizes path similarity by taxonomy depth (dep). This measure is computed as follows.

$$sim_{lc}(C_1, C_2) = -\log\left(\frac{dis(C_1, C_2)}{2 \times dep}\right) \tag{6.7}$$

(3) **Wu–Palmer similarity** is computed based on the depth of two senses ($dep(C)$) in a taxonomy and the depth of a Least Common Subsumer (LCS) (i.e., the lowest node in the hierarchy that is a hypernym of both senses) as follows.

$$sim_{wup}(C_1, C_2) = \frac{2 \times dep\big(LCS(C_1, C_2)\big)}{dep(C_1) + dep(C_2)} \tag{6.8}$$

(4) **Resnik similarity** relies on the concept of Information Content (IC) defined as $IC(C)=-logP(C)$ where $P(C)$ is a probability of encountering an instance in a corpus. This similarity is computed as follows.

$$sim_{res}(C_1, C_2) = -\log P\big(LCS(C_1, C_2)\big) \tag{6.9}$$

(5) **Lin similarity** integrates the concept of IC with similarity theorem as follows.

$$sim_{Lin}(C_1, C_2) = \frac{2 \times \log P\big(LCS(C_1, C_2)\big)}{\log P(C_1) + \log P(C_2)} \tag{6.10}$$

(6) **Jiang–Conrath similarity** exploits IC to define the distance between two concepts as follows.

$$sim_{jc}(C_1, C_2) = \frac{1}{2 \times \log P\big(LCS(C_1, C_2)\big) - \big(\log P(C_1) + \log P(C_2)\big)} \tag{6.11}$$

(7) **ConceptNet** similarity is based on a semantic concept similarity provided by ConceptNet.

(8) **Continuous vector space semantic similarity** is computed using cosine similarity between two word vectors as follows.

$$sim_{cos}(C_1, C_2) = \frac{\vec{\mathbf{V}}(C_1).\vec{\mathbf{V}}(C_2)}{|\vec{\mathbf{V}}(C_1)| \times |\vec{\mathbf{V}}(C_2)|} \tag{6.12}$$

6.4.3 Taxonomy Extension

There are two cases that cannot be addressed by unsupervised taxonomy merging including missing categories and fine–grained categories. Missing categories are categories that do not match WordNet classification (e.g., *Xbox game console*) and hence are misclassified. Fine–grained categories are categories that only their hypernym category is available in WordNet (e.g., WordNet contains a category for a *washing machine* but not for *Front Load Washers*). Unsupervised merging classifies these categories to their parent category, which in turn introduces noise to the model. As an example, both *upright vacuums* and *robotic vacuums* (Roomba vacuum) are fine–grained categories missing from WordNet and are categorized as a *vacuum*. Fine–grained categories tend to have different size and weight distributions in comparison with their siblings.

To address missing categories, we train a logistic regression model. This model receives highest achieved semantic similarity between a missing category and WordNet categories as its

input and decides whether to merge the categories or not. In case that the model rejects the merging, the local hierarchy is added to WordNet categories as a new category. This model is as follows.

$$P(x) = \frac{1}{1 + \exp\left(-\left(w_0 + w_1 \times \max_{i=1,\ldots,N} \left(Sim(\Psi_R^i, \Psi_C) \right) \right) \right)} \tag{6.13}$$

To address fine–grained categories, we use the following heuristic. We collect categories that: (1) are harvested from the same resource; (2) are mapped to the same category in WordNet; (3) have a same immediate ancestor in their local hierarchy; and (4) have different leaves in their local category. We then extend the corresponding WordNet category with more fine–grained child categories by appending the leaves of a local category to WordNet.

6.5 Gaussian Models

We train Gaussian Mixture Models (GMM) to model the underlying distributions of physical attributes of collected objects and then use a trained model to perform probabilistic inference and data sampling. A GMM is defined as a weighted sum of *M* multivariate Gaussian distributions as follows.

$$P(x) = \sum_{i=1}^{M} w_k P(x \mid \mu_i, \Sigma_i) \tag{6.14}$$

M denotes the number of clusters (i.e., the number of Gaussian distributions). w_i, $i=1,\ldots,M$ are the mixture weights (i.e., $\sum w_i=1$). $x=[x_1 \ldots x_n]^{\mathrm{T}}$ is a continuous random variable of physical attributes (i.e., *width*, *length*, *height*, and *weight*) of instances of each category. $P(x|\mu,\Sigma)$ denotes a multivariate Gaussian distribution defined as follows.

$$P(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} \mid \Sigma \mid^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \tag{6.15}$$

μ and Σ denote mean vector and the covariance matrix, respectively. A GMM is parameterized by mean vectors, covariance matrices, and mixture weights. The number of models (*M*) is a hyper–parameter. Using a grid search strategy in a range of [2,15], we set it to 5.

GMM parameters are estimated from training data using iterative Expectation–Maximization (EM) algorithm [167].

We train a GMM per category (each GMM has four attributes) in the constructed taxonomy and a GMM per model category in ShapeNet to create an ensemble of two models. GMMs trained on ShapeNet learn distributions of dimension ratios (i.e., *length/width*, *width/height*, etc.). These models are then exploited to reduce the sampling noise of GMMs trained on the web data using ranking selection. Given an object of interest, we first sample a corresponding GMM on object's category for k times and then feed these samples to a corresponding GMM trained on dimension ratios and compute the probability of each sample. The most probable sample is then selected as the result. This technique reduces the effect of the noise learned by the categorical GMMs.

6.6 Experimental Results

To model physical attributes of objects, we considered three assumptions: (1) underlying distributions are Gaussian or multimodal; (2) a mixture of distributions fits the data better; and (3) physical attributes are not independent. Due to noisy nature of the web data, we visualized data histograms (i.e., with 50 bins) rather than using systematic goodness–of–fit tests such as Kolmogorov–Smirnov test to investigate the underlying distributions. As an instance, a 50–bin data histogram for *men's bag* category including 523 instances is depicted in Figure 6.2. As shown, distributions are either Gaussian or multimodal. We observe similar behaviors for other categories as well and hence data visualization validates the first assumption.

The noisy nature of data also implies that simple Gaussian distributions cannot fit it well and hence it is required to use a mixture of such models (as shown in Figure 6.2). The third assumption can be validated both analytically (i.e., density is defined as $\rho=m/v$) and empirically (e.g., Pearson correlation coefficient and the p–value for volume–weight pair are 0.6392 and 4.58E–08, respectively).

To investigate the performance of enriching the taxonomies, we sampled 200 categories and defined 10 gold standard phrases for each category. We then applied the mentioned feature selection methods on this dataset. The results are shown in Table 6.2. Also, the McNemar's test results are shown in Table 6.3.

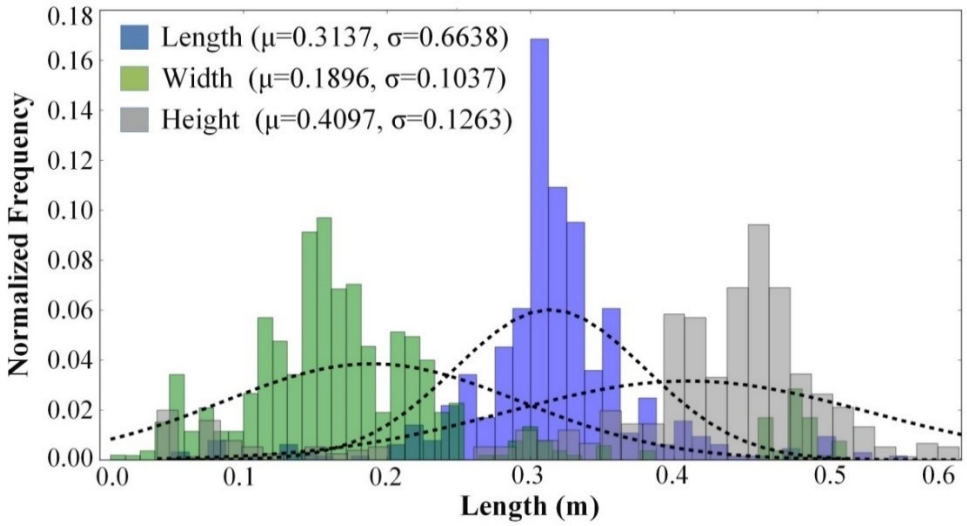

Figure 6.2. A Sample distribution of size attributes of an object. Distribution is extracted from 523 instances of *men's bag* dimensions. All dimensions follow either Gaussian or multimodal distributions

Table 6.2. The accuracy of feature selection methods. Normalized TF–IDF method slightly outperforms other feature selection techniques

	MI	**χ^2**	**TF–IDF**
Mean	88.1%	87.9%	89.2%
SD	2.53	4.32	2.89

Table 6.3. Results of the McNemar's statistical test on the feature selection methods

Feature selection pair	**χ^2 estimation**	**P**
TF–IDF ⊗ MI	3.844	<0.05
TF–IDF ⊗ χ2	3.924	<0.05

Results suggest that normalized TF–IDF slightly outperforms other methods. The error analysis showed that unigrams are a frequent source of error (e.g., for "*air purifiers"* category all methods select "*air"* as a candidate). Hence, it is necessary to prioritize the candidates with respect to their lengths. Also, the McNemar's test results suggest that the differences in results are statistically significant.

We semi–automatically assigned the sampled 200 categories with WordNet hypernym hierarchies and manually validated them, and then used it to compare the performance of

similarity measures. Results are shown in Table 6.4. In terms of accuracy, precision, and F_1–score, Jiang–Conrath WordNet–based similarity outperforms other measures whereas in terms of recall, ConceptNet–based similarity is the dominant approach. Results also suggest that lexical similarities outperform both continuous word vector similarities. Also, the McNemar's test results shown in Table 6.5 shows the statistical significance of the results.

Table 6.4. Results of the McNemar's statistical test on the semantic similarity measures

Similarity	Accuracy	Precision	Recall	F_1–score
Binary	58.50%	0.5812	0.5220	0.5500
Path	78.00%	0.7925	0.7812	0.7868
Leacock–Chodorow	79.50%	0.7786	0.7624	0.7704
Wu–Palmer	88.00%	0.9054	0.8856	0.8954
Resnik	92.00%	0.9185	0.8928	0.9055
Lin	93.00%	0.8964	0.9010	0.8987
Jiang–Conrath	**94.50%**	**0.9486**	0.9158	**0.9319**
ConceptNet	92.50%	0.9187	**0.9358**	0.9271
Word2Vec	85.00%	0.8624	0.8245	0.8430
GloVe	85.50%	0.8858	0.8425	0.8636

Table 6.5. Performance of semantic similarity measures. Jiang–Conrath WordNet–based similarity outperforms other models in terms of accuracy, precession, F_1–score, whereas ConceptNet similarity model performs better in terms of recall

Similarity pair	χ^2 estimation	P
Jiang–Conrath ⊗ Binary	100.084	<0.005
Jiang–Conrath ⊗ Path	45.831	<0.005
Jiang–Conrath ⊗ Leacock–Chodorow	44.485	<0.005
Jiang–Conrath ⊗ Wu–Palmer	13.793	<0.005
Jiang–Conrath ⊗ Resnik	9.000	<0.005
Jiang–Conrath ⊗ Lin	8.533	<0.005
Jiang–Conrath ⊗ ConceptNet	10.563	<0.005
Jiang–Conrath ⊗ Word2Vec	10.125	<0.005
Jiang–Conrath ⊗ GloVe	15.158	<0.005

To evaluate the performance of the proposed model on the taxonomy extension, we trained the logistic regression model defined in (6.13) on 200 categories (50 missing from WordNet) and tested it on 50 categories (25 missing from WordNet). The model achieved an

accuracy of 94% with F_1–score of 0.9214. This suggests that a linear classifier can address the taxonomy extension.

To evaluate the overall performance of the proposed approach, we performed two scenarios: (1) automatic scaling of 3D models, and (2) answering comparative questions. The first scenario is a subjective scenario in which trained models are exploited to automatically scale a set of 3D objects. We generated 10 scenes with 12 scaled objects in each scene and asked four users to score the naturalness of relative sizes based on the Likert scale (i.e., 1 represents not natural and 5 represents very natural). In 87.5% of cases, users ranked the scenes as very natural and in 12.5% of cases they ranked them as natural. A sample scene scaled by trained GMMs is shown in Figure 6.3. Figure 6.4 shows a set of 3D models in their default sizes and same models after being automatically scaled using the proposed approach.

We also evaluated simple inference capabilities of trained models. For this purpose, we defined a set of comparative questions using grammar G as follows.

$$\begin{aligned} &G : S \rightarrow is\ X\ C\ than\ Y\ ? \\ &C \rightarrow smaller \,|\, bigger \,|\, lighter \,|\, heavier \end{aligned} \tag{6.16}$$

To infer answers, we utilize Monte Carlo simulation [196]. We sample GMMs corresponding to categories of X and Y, and compare a sample pair with respect to C. We repeat this process for N=10,000 times and count the number of pairs (K) that satisfy $C(X,Y)$. Answers are generated as follows:

(1) X is C than Y with confidence of k/N, if $k>>N/2$

(2) X is not C than Y with confidence of $1–k/N$, if $k<<N/2$

(3) X is Similar to Y with confidence of $2\times k/N$, if $k\approx N/2$

We manually constructed a dataset of 500 pairs such as <*guitar, bigger, spoon*> and <*refrigerator, heavier, teapot*>, and asked the system to accept or reject these assertions. It correctly accepted 94.6% of them. Rejections were generated in cases that both GMMs corresponding to categories of X and Y were trained on less than 100 instances. It is noteworthy that average training time per category is 57 milliseconds and average sampling time for 10,000

samples is 5 milliseconds on an Intel Core i7 processor. This suggests that the proposed method is both real–time and efficiently scalable.

Figure 6.3. A sample scene scaled by trained GMM models and represented to four users to evaluate the naturalness of relative sizes based on the Likert scale

We addressed the collecting and mapping process of the web data from heterogeneous resources into the WordNet taxonomy using a normalized semantic similarity measure. We also trained a logistic regression model to validate the mapping process. Finally, we trained Gaussian mixture models to learn the distributions of the *size* and *weight* data from the collected data. The results suggest that training unsupervised models on raw data and then validating them using supervised models achieve promising performance.

Figure 6.4. (a) Offline–learning to assign absolute sizes to 3D models. The upper row demonstrates a collection of 3D models in their default sizes; Same models are shown in the lower row after being automatically scaled using the offline–learning. Size distributions of objects are learned from various retail store websites such as Walmart, IKEA, etc.

Chapter 7. Online–Learning Physical Attributes

Online–learning is designed to cover the missing knowledge that cannot be retrieved using offline–learning. It can predict attributes and their values of an unseen object in an online manner. The goal is to find which attributes are feasible for an object, what their corresponding values are, and what the underlying parameters are.

Schematic of online–learning approach shown in Figure 7.1 is a pipeline of three components; attribute selector predicts plausible physical attributes of a given object using a random forest classifier; value extractor predicts whether a numeric value within an HTML document corresponds to a specific object–attribute pair. It uses a deep Convolutional Neural Network (CNN) model for this purpose; Distribution estimator uses Maximum Likelihood Estimation (MLE) to model extracted values as Gaussian distributions.

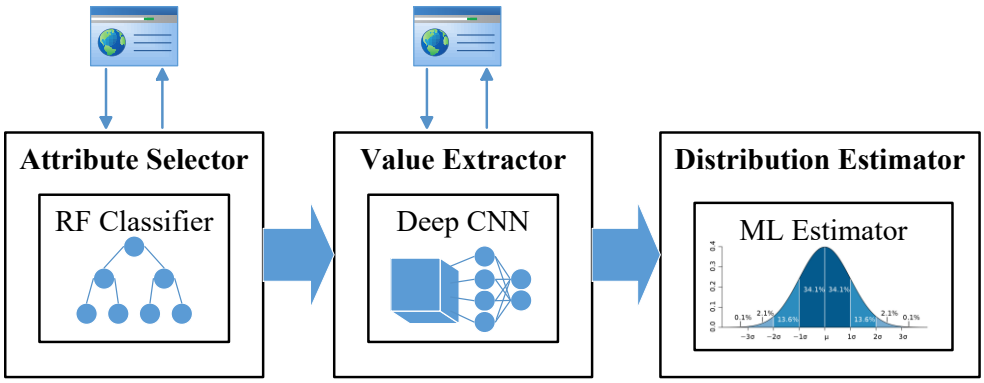

Figure 7.1. Schematic of online–learning approach. Attribute Selector predicts plausible attributes for a given 3D model (e.g., length, weight, and maximum speed for a *rabbit*), Value Extractor extract potential attribute values (e.g. weight of a *rabbit* can be 0.7 kg, 1.8 kg, 1.1 kg, etc.) and Distribution Estimator produces a distribution of attribute values (e.g. weight of a *rabbit* follows Gaussian distribution with mean value of 1.2 kg and a standard deviation of 0.5kg)

7.1 Problem formulation

We use Object–Attribute–Value (OAV) triplets as the knowledge representation scheme. The OAV formalism is a specialized case of semantic networks in which object–attribute and attribute–value relations are constrained to *HAS–A* and *IS–A* relations, respectively. This formalism represents an association between object $o \in O$ and value $v \in V$ of attribute $a \in A$ (i.e., an attribute $a \in A$ is considered as a function $a: O \rightarrow V$). To address variations in measurements, precision of unit conversions, and diversity of objects, we extend the OAV representation to Object–Attribute–Distribution (OAD) formalism in which a value of an attribute is replaced by a distribution over possible values of that attribute (i.e., under the assumption of normal distribution for attribute a, we have: $a: O \rightarrow N(\mu,\sigma)$). As an example, given an *apple tree* as an object of interest, one can model its height using the following OAD triplet <*apple tree, height, N~* (*200cm,50cm*)>.

Given this representation, the problem can be formalized as finding a set of attributes, values, and an underlying distribution for a given object such that it maximizes the joint probability of $P(O,A,V,D)$. This probability can be further expanded using the chain rule as follows:

$$\arg\max_{O,A,V,D} P(O,A,V,D) = P(O) \times P(A \mid O) \times P(V \mid A,O) \times P(D \mid O,A,V) \tag{7.1}$$

$P(O)$ is a probability of selecting object O as query object. Assuming a uniform distribution for selecting an object, this probability is equal for all objects and hence does not affect the maximization process. $P(A|O)$ is the probability of object O having attribute A as its physical attribute (e.g., the *area* is a common attribute of a building but not a human). To maximize this probability, we utilize the web data to train a classifier which can predict physical attributes of a query object. $P(V|A,O)$ indicates the likelihood of a value V given object O and attribute A. As an example, given an *apple tree* as an object of interest and *height* as its attribute, probability of value 190cm is very high, whereas the likelihood of 10cm is very low. It is noteworthy that these values are not available in advance. To address this, we first retrieve top k=10 documents containing candidate values from the web and then train a classifier to predict whether a retrieved value is associated with a given attribute of a query object or not.

$P(D|O,A,V)$ denotes the likelihood of an underlying distribution being of type D. To simplify the learning process, it is assumed that all attributes follow a similar distribution. To decide the distribution type, real–world data is collected from the web and analyzed. Results, as discussed in chapter 6, suggest that Gaussian distribution fits the data best. Hence, it is assumed that: $D \rightarrow N \sim (\mu, \sigma)$ for all attributes. Therefore, by assuming a uniform distribution for object selection and a normal distribution for attributes, the problem boils down to maximizing the joint probability by finding appropriate attributes and values for a query object O as follows.

$$\arg\max_{A,V} P(O, A, V, D) = P(A \mid O) \times P(V \mid A, O) \tag{7.2}$$

7.2 Related Work

In [11], a set of meronyms patterns is utilized to extract attributes of a given concept from the web. Attributes are then ranked using Google web hit count as: *score*(*attribute*) = *hit*(*attribute* & *concept*) / *hit*(*attribute*). A similar approach is used to rank the values associated with each attribute. Finally, a threshold is used to keep top attributes and values to construct ontology. While our approach uses a similar technique, it does not require a predetermined threshold. Also, due to noise embedded within the web content, we use non–linearity for better generalization in attribute selection. A similar approach is used in [197] which amalgamates instance–based and concept–based attribute selection methods using Bayes' theorem. In [13], query templates and point–wise mutual information are used to extract attributes from a set of web search queries.

In [198], semantic entropy is utilized to extract relevant attribute–value pairs from product descriptions across different websites. The structural–semantic entropy is used to locate the data on web pages. It measures the density of occurrences of relevant information on a Document Object Model (DOM) tree. This approach is restricted to specific web structures and cannot address sparse web documents. In [199], a set of linguistic clues (e.g., holonymy relations) are employed to extract object sizes. It represents a size as a linear function of object features and trains a regression and ranking model to approximate this function. The system reaches an accuracy of 80%. In [12], a set of hand–crafted patterns such as *Object is* * [*height unit*] *tall* or *Object width is* * [*width unit*] are used to extract and approximate numerical object

attributes such as *height* and *weight* from the web. This approach cannot generalize to unpredicted attributes.

In [200], unique entity identifiers (e.g., ISBN, GTIN, DOI) are systematically harvested from the web to construct a knowledge–base of such information. The studied method segments a web page into a frame–tree and utilizes a distance heuristic to select a set of identifiers. In [201], a rule–base is used to extract measured information from textual documents. Rule–base contains a large set of both common and obscure measurement units. SCAD [202] is a system designed to expand and complete product catalogs in e–commerce websites by leveraging signals from web page context and a collective analysis of all pages corresponding to an entity. It uses Integer Linear Program (ILP) to integrate signals into decisions over attributes.

The superiority of our approach to these works is that it learns to predict which attributes are plausible for an object, and which numeric values are associated with those attributes. Hence, it does not require any pre–determined patterns or thresholds and can address unseen object–attribute pairs.

7.3 Attribute Prediction

Attribute selection is used to predict a set of plausible physical attributes of a given object in a way that prediction maximizes a probability of observing those attributes within the web content. For instance, although *volume* and *diameter* are valid physical attributes for objects, yet it is very unlikely to observe them for *bears* or *humans* on the web. As another example, *speed* is a common physical attribute of all animals and its corresponding values for many animals such as a *bee*, *tiger*, *elephant*, *eagle*, etc. and athletes such as *Christiano Ronaldo* are available online. However, this information is not available for animals such as *pangolin*, *Alaskan malamute, woodlouse spider* or celebrities such as *George Clooney*.

Unsupervised attribute selection is an appropriate technique for this purpose [11], but is very sensitive to threshold measures (i.e., decision boundary). For instance, one can define a score such as $S(O,A)=Count(O\cap A)/Count(O)$ to measure associations between object O and attribute A, and define a threshold ζ to select those attributes with $S(O,A)\geq\zeta$. Nevertheless, deciding ζ is not a trivial task and significantly affects the performance. Furthermore,

unsupervised models assume that the relation between these two counts is linear which is not a valid assumption.

To alleviate this issue, rather than hand–crafting thresholds, we train a binary classifier to predict whether an attribute is plausible for a given object or not. We pre–determine a set of physical attributes that are important in 3D worlds including possible attributes of size (*length, height, width, depth, thickness, radius,* and *diameter*), speed (*speed, velocity*, and *acceleration*), and weight (*weight* and *mass*). This classifier predicts a subset of pre–determined attribute set as plausible attributes of an object of interest in a way that it maximizes $P(A|O)$.

We use a Random Forest (RF) classifier for this purpose. Input features of the classifier are a pair of a Google web hint count of an object (e.g., the number of search results for *apple tree*) and a joint Google web hint count of an object–attribute pair (e.g., the number of search results for *apple tree height*). Output is either 0 (implausible attribute) or 1 (plausible attribute). For an object of interest, hit count pairs are fed to the trained classifier for all candidate attributes and then attributes predicted as implausible are filtered out.

7.3.1 Web Count Dataset

We collected a dataset of 2,572 objects (including animals, fruits, celebrities, mountains, towers, and furniture) by crawling multiple websites (e.g., a–z–animals.com, imdb.com, etc.). We also recruited workers on Amazon Mechanical Turk (AMT) and asked them to determine the plausibility of an object–attribute pair based on top 10 web documents retrieved by Google's exact phrase search results. We then used Google search engine API to collect pairs of web hit counts. The final dataset consists of 15,414 labeled samples from which 5,713 are positive samples (i.e., plausible attribute) and 9,701 are negative. We used stratified sampling to split the dataset to a training dataset of 14,000 samples (5,189 positive and 8,811 negative), and a test dataset of 1,414 (524 positive and 890 negative) samples.

7.4 Attribute Value Prediction

Next step is to assign numeric values to predicted attributes. We train another classifier that predicts whether a potential numeric value within an HTML document is associated with an object– attribute pair or not. This classifier maximizes $P(V|O,A)$.

7.4.1 Data Collection and Preprocessing

We asked recruited workers on AMT to collect minimum and maximum values of plausible attributes of the collected dataset based on top 10 Google results (e.g., given *tiger weight* as object–attribute pair, Google search results in: *adult male tiger: 90–310 kg*; *adult female tiger: 65–170 kg* and hence the minimum and maximum weight for a *tiger* is set to 65 kg and 310 kg, respectively).

We then used Google search engine API to collect top 10 HTML documents for all plausible object–attribute pairs (i.e., a dataset of 43,630 HTML documents). HTML documents carry many irrelevant numeric data which can dramatically degrade the performance of a classifier. To address this issue, we developed a set of 15 regular expressions to capture measurement numeric data with different typesets and units (e.g., *length*: 3 1/2 inches, 2'3", 2/10 cm; *weight*: .02 kg, 2×10^2 tones; *velocity*: 2 km/hr, 1.2 mph, etc.) within inner text of 49 types of HTML tags. Some of the utilized tags are shown in Table 7.1.

Table 7.1. Informative HTML tags. A subset of HTML tags used to extract candidate attribute values from an HTML document

Tag	HTML Structure
Title	*<title> Text </title>*
Header 1	* Text *
Header 2	* Text *
Header 3	* Text *
Meta	*<meta name = "descriptions" content = Text/>*
Meta	*<meta name = "keywords" content = Text/>*
Image	**
Table	*<td> Text </td>*
Paragraph	* Text *
Hyperlink	* Text *
Font	* Text *
Bold	* Text *
Italic	* Text *
Underline	*<u> Text </u>*

We then filtered out numeric data whose units are inconsistent with a query attribute. As an instance, given that a document is retrieved for query: *queen mattress width,* all numeric

values that are matched by *weight* and *speed* regular expressions are ignored. Remaining numeric values are then converted to the SI base units.

After locating a set of candidate numeric values, Document Object Model (DOM) tree of an HTML document is used to extract a set of primitive features. For each numeric value, we extract HTML tags of the parent, ancestor, and left and right siblings of the text that contains the value. We also extract a context window of length $2k + 1$ by tokenizing surrounding text of a numeric value, centering window on the value, and selecting k terms to the left and k terms to the right of the value. If surrounding text of either side is less than k terms, we pad it using <*PAD*> symbol. We also use a special symbol <*SW*> to indicate that the text within a context window belongs to more than one HTML tags. An extracted context window along with extracted ancestors' and siblings' tags and corresponding object–attribute pair are concatenated and considered as a training sample. A class of each sample is assigned as follows.

Extracted and pre–processed numeric values for each object–attribute pair from HTML documents are compared against the minimum and maximum values of that pair from the dataset collected by AMT workers. If a numeric value is within the range of minimum and maximum values with a margin of ±2% (i.e., to compensate for conversion or human errors), the sample is labeled as 1 (i.e., positive), otherwise 0 (i.e., the numeric value is not associated with the object–attribute pair).

Finally, numeric data is replaced by <*NUM*> symbol. For example, given a query: *Madonna's height*, and having an entry in AMT dataset as *Madonna's height: Min*=1.71*, Max*=1.74, and assuming $k = 6$, a feature vector extracted from the following HTML is: *F:*<[*Madonna, height, <p>, , <h1>, <h2>, Information, <SW>, Her, height, is, about, <NUM>, meters, <SW>, Weight, Information, <PAD>, <PAD>*]^T, [+*1*]>.

```
<div>
   <h1> Height Information </h1>
   <p> Her height is about 1.73 meters </p>
   <h2> Weight information </h2>
</div>
```

This results in a dataset of 66,685 instances (28,845 positive and 37,840 negative). We further split the dataset to a training set with 60,000 samples (25,955 positive and 34,045

negative), and a test set with 6,685 (2,890 positive and 3,795 negative) using stratified sampling. Using this representation, we formulate the problem of value extraction as a binary classification task in which given a query (object–attribute pair), local HTML structure, and surrounding text, a classifier is trained to learn the underlying associations to predict whether a value within a retrieved HTML document is an answer to an unseen query. Nevertheless, primitive feature representation is not adequate to train a classifier. Hence, we augment the extracted dataset by engineering various feature categories.

7.4.2 Feature Engineering

We defined a primitive feature as $[\mathbf{Q\ S\ C}]^T$ where Q denotes a query (object–attribute pair), S denotes a structure (parent's, ancestor's, and siblings' HTML tags), and C denotes a context window. To enhance the classification performance, we augment the textual content of a primitive feature (Q and C) with four different feature sets including linguistic features, pre–trained universal word embeddings, corpus–trained word embeddings, and hybrid features to decide the best feature representation scheme. These features are as follows.

Linguistic Features

Linguistic features consist of nine features including five lexical features, one syntactic feature, and three semantic features. For a given linguistic content $[w_{i-k}, \ldots, w_i, \ldots, w_{i+k}]$, lexical features are as follows.

1. Unigrams:

$$\mathbf{\Phi_1}: [w_{i-k}, \ldots, w_i, \ldots, w_{i+k}] \tag{7.3}$$

2. Lemmas:

$$\mathbf{\Phi_2}: [l_{i-k}, \ldots, l_i, \ldots, l_{i+k}] \tag{7.4}$$

3. 1–skip–bigrams:

$$\mathbf{\Phi_3}: [w_{i-k}\ w_{i-k+1},\ w_{i-k}\ w_{i-k+2}, \ldots, w_i\ w_{i+1}, \ldots,\ w_{i+k-1}\ w_{i+k}] \tag{7.5}$$

4. Frequency score of unigrams:

$$\mathbf{\Phi_4}: [f(w_{i-k}), \ldots, f(w_i), \ldots, f(w_{i+k})] \tag{7.6}$$

5. Frequency score of 1–skip–bigrams:

$\mathbf{\Phi_5}$: $[f(w_{i-k+1}|w_{i-k}), \ldots, f(w_{i+1}|w_i), \ldots, f(w_{i+k}|\ w_{i+k-1})]$ (7.7)

Unigram and lemmas are extracted by tokenizing and lemmatizing the textual content using CoreNLP [29], and 1–skip–bigrams are extracted using NLTK toolkit [163]. We define frequency score of a word *w* as follows:

$$f(w) = Ntf.idf(P,w) - Ntf.idf(N,w) \tag{7.8}$$

Ntf.idf (*P,w*) and *Ntf.idf* (*N,w*) are normalized TF–IDF scores of *w* occurring in positive and negative examples, respectively. *Ntf.idf* (C,*T*) is computed as follows.

$$Ntfidf(C,T) = \left(\gamma + (1-\gamma)\frac{tf(C,T)}{tf_{\max}(C)}\right) \times \log\left(1 + \frac{|C|}{N_T}\right) \tag{7.9}$$

tf(*C,T*) is frequency of term *T* in category *C*, $tf_{max}(C)$ is maximum frequency of a term in category *C*, |*C*| is the number of classes (i.e., two in this case), N_T is the number of classes that *T* appears within, and *g* is a smoothing term set to 0.4 as advised by [194]. The utilized syntactic feature is as follows:

6. Word–level Part–Of–Speech (POS) tags:

$\mathbf{\Phi_6}$: $[pos(w_{i-k}), \ldots, pos(w_i), \ldots, pos(w_{i+k})]$ (7.10)

This feature is extracted using CoreNLP POS–tagger [30]. Three semantic features are as follows:

7. Named–entities:

$\mathbf{\Phi_7}$: $[ne(w_{i-k}), \ldots, ne(w_i), \ldots, ne(w_{i+k})]$ (7.11)

8. An indicator to decide whether a word refers to an abstract or a physical entity:

$\mathbf{\Phi_8}$: $[phy(w_{i-k}), \ldots, phy(w_i), \ldots, phy(w_{i+k})]$ (7.12)

9. Polysemy score of words:

$\mathbf{\Phi_9}$: $[pol(w_{i-k}), \ldots, pol(w_i), \ldots, pol(w_{i+k})]$ (7.13)

Named–entities are extracted using Stanford named–entity recognizer [35]. Features $\mathbf{\Phi_8}$ and $\mathbf{\Phi_9}$ are extracted using WordNet [19]. Finally, the linguistic feature is defined as a concatenation of these nine features as follows.

$\mathbf{F_L}$: $[\mathbf{\Phi_1}\ \mathbf{\Phi_2}\ \mathbf{\Phi_3}\ \mathbf{\Phi_4}\ \mathbf{\Phi_5}\ \mathbf{\Phi_6}\ \mathbf{\Phi_7}\ \mathbf{\Phi_8}\ \mathbf{\Phi_9}]^T$ (7.14)

Universal Word Embeddings

We consider two word embedding models including Word2Vec and GloVe models.

$$\mathbf{F_{W2V}}: [H_{w2v}(w_{i-k}), \ldots, H_{w2v}(w_i), \ldots, H_{w2v}(w_{i+k})] \quad (7.15)$$

$$\mathbf{F_{GloVe}}: [H_G(w_{i-k}), \ldots, H_G(w_i), \ldots, H_G(w_{i+k})] \quad (7.16)$$

$H_{w2v}(w_k)$ and $H_G(w_k)$ are hash functions that retrieve corresponding vectors of a given word. Missing words in both pre–trained models are randomly generated by sampling each dimension from distribution: U~[–1,+1].

Local Word Embeddings

We also train local word embeddings on a corpus of collected HTML repository with 1,733,538 tokens and 59,814 words based on Word2Vec model. Training is performed using genism semantic modeling library [164]. This feature is defined as follows.

$$\mathbf{F_{Gen}}: [H_{Gen}(w_{i-k}), \ldots, H_{Gen}(w_i), \ldots, H_{Gen}(w_{i+k})] \quad (7.17)$$

$H_{Gen}(wi)$ is a hash function that retrieve a locally–trained vector of a given word.

Hybrid Features

We also define a hybrid feature by combining universal word embeddings with linguistic features. For this purpose, we replace lexical features with their corresponding word embeddings as follows.

$$\mathbf{F_{Hybrid}}: [\mathbf{F_2\ \Phi_6\ \Phi_7\ \Phi_8\ \Phi_9}]^T \quad (8.18)$$

Using these feature sets, we augment a primitive feature (i.e., $[\mathbf{Q\ S\ C}]^T$) as follows.

$$\mathbf{F_1}: [\mathbf{F_L(Q)\ S\ F_L(C)}]^T \quad (8.19)$$

$$\mathbf{F_2}: [\mathbf{F_{W2V}(Q)\ S\ F_{W2V}(C)}]^T \quad (8.20)$$

$$\mathbf{F_3}: [\mathbf{F_{GloVe}(Q)\ S\ F_{GloVe}(C)}]^T \quad (8.21)$$

$$\mathbf{F_4}: [\mathbf{F_{Gen}(Q)\ S\ F_{Gen}(C)}]^T \quad (8.22)$$

$$\mathbf{F_5}: [\mathbf{F_{Hybrid}(Q)\ S\ F_{Hybrid}(C)}]^T \quad (8.23)$$

We decide a final feature set by performing comprehensive experimental results on these feature sets.

7.5 Attribute Distribution Estimation

Due to precision limitations, variations in measurements, subjective judgments, unit conversions, granulation levels, and intra–variations in object categories, there is always more than one correct value for a given attribute of an object on the web. As an example, given "*elephant"* as a query object, different species and genders of elephants show different values for physical attributes. To address this, we estimate an underlying distribution of such information and then sample it to assign a value for an attribute of interest. Also, because of sampling variance, this method results in natural variances in 3D models (e.g., having two elephant models in a scene with slightly different sizes is more realistic than two elephants with exact same sizes). We model the extracted values as a normal distribution as follows.

$$p(v_k \mid O, A, D) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(v_k-\mu)^2}{2\sigma^2}} \tag{7.24}$$

We utilize Maximum Likelihood Estimator (MLE) to estimate the parameters of a distribution (i.e., μ and σ) as: $\mu = \sum_{i=1}^{|V|} v_i \big/ |V|$ and $\sigma = \sqrt{\sum_{i=1}^{|V|} (v_i - \mu)^2 \big/ |V|}$, where $|V|$ is the number of extracted values for an attribute.

7.6 Experimental setup

7.6.1 Learning Models

Similar to spatial role labeling task, we utilize four linear models (logistic regression, k–nearest neighbor (K–NN), linear support vector machine (SVM), and Bernoulli naive Bayes classifiers), three ensemble models (random forest, Ada–boost, and bagging classifiers) (Appendix F), and two deep learning models (fully–connected feed–forward deep neural network (DNN) and deep convolutional neural network (CNN) classifiers) for this task. Linear and ensemble models are applied using scikit–learn library [167] whereas deep models are implemented using TensorFlow library [168].

7.6.2 Models Setup

Both DNN and CNN models are trained using Adam stochastic optimizer [169] with a learning rate of 1E–4 over the mini–batches of size 250. Mini–batches are uniformly sampled with replacement from the training set. Both models utilize a cross–entropy loss function with one–hot output representation. Models use dropout regularization [170] with a probability of p=0.5 and batch normalization [171] on input layer. Both models also utilize Rectified Linear Units (ReLU) in their hidden layers and a softmax function in their output layer. DNN model is defined as a four–layer network (3 hidden layers + softmax layer). CNN architecture shown in Figure 7.2 consists of two convolutional layers each followed by a max pooling layer and two fully–connected layers followed by a soft–max output layer.

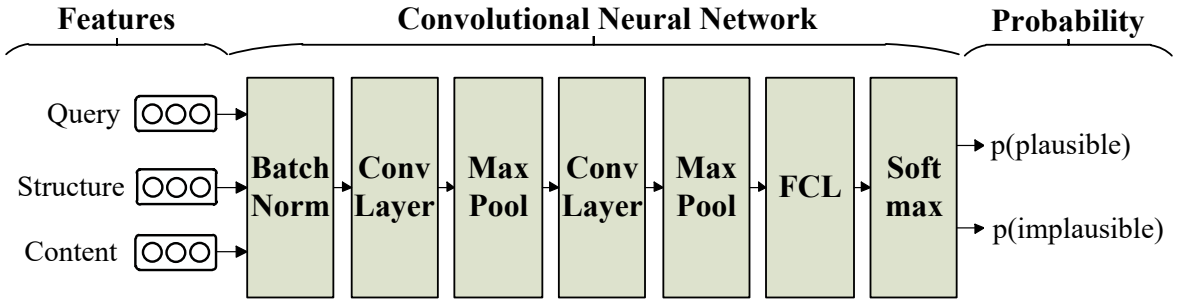

Figure 7.2. Schematic of proposed CNN model for attribute value extractions. Inputs are combination of query, HTML structure, and content features, and output is either 0 (value is not associated with query) or 1 (value is the answer to query)

We trained networks with 50,000 epochs and used 32, 64, and 128 filters with sizes of 2×5, 3×5, and 5×5 with a stride of 1. We also used windows of sizes 1×3, 2×2, 3×2 and 3×5 for pooling. Sizes of hidden layers are considered as hyper–parameters and are optimized using a random search strategy. For this purpose, 10% of train set is sampled as development (validation) set. For different feature sets, these sizes are set to 800, 800, and 1,000 for DNN, and 800 and 600 for CNN respectively. Hyper–parameters of linear and ensemble models are set by sampling the hyper–parameter spaces for 200 times and 10–fold cross–validation.

7.7 Experimental Results

7.7.1 Attribute Prediction

Considering the low dimensionality of this task, we only utilized linear and ensemble models and compared the results with an unsupervised approach introduced in [11]. Classification results and the McNemar's test results are shown in Tables 7.2 and 7.3, respectively.

Table 7.2. Performance of classifiers and an unsupervised model on attribute prediction task. Random forest classifier outperforms other models in terms of precision, recall, and F_1-score

Classifier	Precision	Recall	F_1–score
Adaboost	0.8622	0.7509	0.8027
Random Forest	**0.9624**	**0.9144**	**0.9378**
Bagging	0.9406	0.8935	0.9165
K–NN	0.8845	0.8020	0.8412
Linear SVM	0.8806	0.7644	0.8184
Logistic regression	0.8988	0.8174	0.8562
Unsupervised [11]	0.6928	0.5945	0.6399

Table 7.3. The McNemar's test results on the attribute prediction task

Classifier pairs	χ^2 Estimation	P
Random Forest ⊗ Adaboost	69.031	<0.005
Random Forest ⊗ Bagging	2.077	<0.1
Random Forest ⊗ K–NN	70.777	<0.005
Random Forest ⊗ Linear SVM	89.689	<0.005
Random Forest ⊗ Logistic regression	89.965	<0.005
Random Forest ⊗ Unsupervised [11]	236.742	<0.005

As shown, random forest classifier outperforms other models in terms of precision, recall, and F_1–score. Also, as suggested supervised models significantly outperform the unsupervised model. Classification errors root in outliers. For instance, given query *q:<Keanu Reeves, speed>*, web hit count favors the plausibility of this object–attribute pair. However, high hit count of this

pair is because of a movie named *speed* played by *Keanu Reeves*. Exploiting a knowledgebase such as ConceptNet [21] or Google knowledge graph can enhance this process.

7.7.2 Value Prediction

We developed five feature sets indicated in (7.19) – (7.23) for attribute value extraction task. We first investigate the effect of context window size on these features. We create features with context window sizes of 5, 11, 15, and 21 words from HTML content and average the accuracy of top two ensemble models (i.e., random forest and bagging classifiers) and two deep learning models. Figure 7.3 shows this effect for all five introduced features.

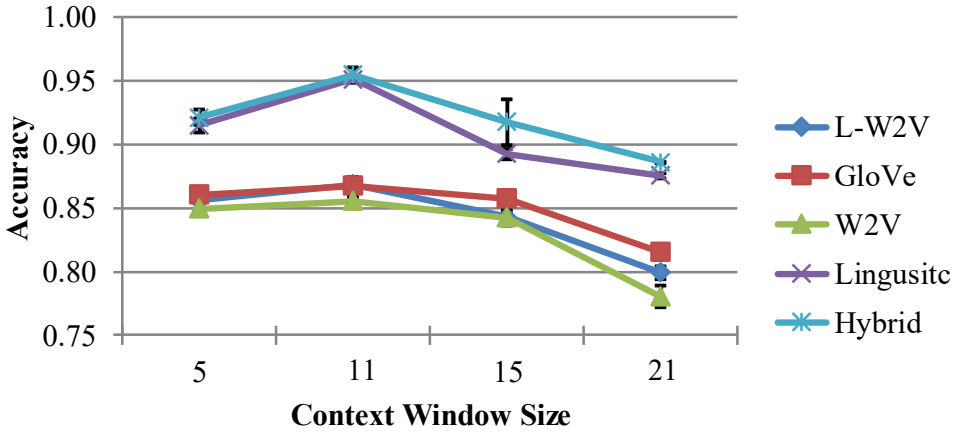

Figure 7.3. Effect of context window size on the average accuracy of random forest, bagging, DNN, and CNN models. All models perform best with a context window size of 11 and proposed hybrid feature

As shown, all feature sets achieve highest average accuracy with context window size of 11 and then it decreases as window size increases. This suggests that an optimal signal to noise ratio is in a local window of 10 words and increasing this window increases the noise proportionally. Also, it is shown that our proposed hybrid feature outperforms other feature sets.

We also investigate the effect of the dimension of word vectors on the performance of value extractor. For this purpose, in addition to 300–dimension vectors, we extract word vectors with 50, 100, 150, and 200 dimensions using Principle Component Analysis (PCA). Similarly,

we average the accuracy of random forest, bagging, DNN, and CNN classifiers on all feature sets except linguistic feature (i.e., because it represents words as unique identifiers, not continuous vectors). Figure 7.4 shows this effect for four introduced features. As shown, best results are achieved with 100–dimension word vectors.

Attribute value prediction dataset has a slight skewness. Hence, accuracy is a more or less valid criterion to evaluate the performance of models on it. We clamp context window size and word vector dimensions to 11 and 100, respectively and train the models on all five feature sets. Results are shown in Tables 7.3–7.7, respectively. Note that due to the symbolic treatment of words in the linguistic feature, we train a naive Bayes classifier using this feature set.

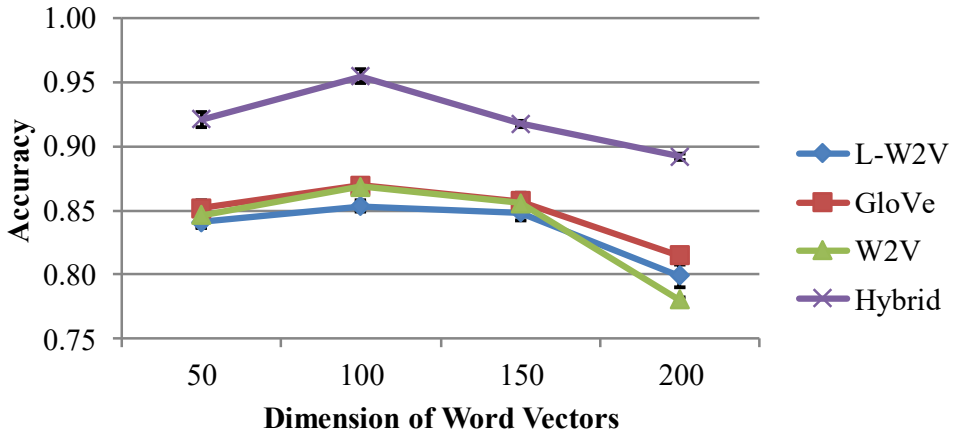

Figure 7.4. Effect of the dimension of word vectors on average accuracy of random forest, bagging, DNN, and CNN models. Best dimension for all feature sets is 100

Table 7.4. Classification results for locally–trained word embeddings. CNN model outperform other models in terms of accuracy, recall, and F_1-score

Classifier	Accuracy	Precision	Recall	F_1–score
Adaboost	77.94%	0.7939	0.6612	0.7215
Random Forest	84.01%	**0.9063**	0.7028	0.7917
Bagging	85.46%	0.8546	0.7574	0.8183
K–NN	83.62%	0.8027	0.8235	0.8130
Linear SVM	78.70%	0.8229	0.6464	0.7240
Logistic regression	76.86%	0.8184	0.5972	0.6905
DNN	85.12%	0.8419	0.8073	0.8242
Proposed CNN	**86.12%**	0.8540	**0.8272**	**0.8404**

Table 7.5. Classification results for pre-trained GloVe word embeddings. DNN model outperform other models in terms of accuracy, recall, and F_1-score

Classifier	Accuracy	Precision	Recall	F_1–score
Adaboost	77.92%	0.7753	0.6889	0.7296
Random Forest	84.82%	**0.8908**	0.7394	0.8081
Bagging	85.98%	0.8880	0.7734	0.8267
K–NN	84.19%	0.8242	0.8062	0.8151
Linear SVM	77.49%	0.7277	0.7657	0.7462
Logistic regression	79.09%	0.7860	0.7093	0.7457
DNN	**86.46%**	0.8541	**0.8284**	**0.8410**
CNN	86.38%	0.8640	0.8180	0.8404

Table 7.6. Classification results for pre-trained W2V word embeddings. Ensemble models perform better in terms of accuracy and precision whereas DNN model archives better results in terms of recall and F_1-score

Classifier	Accuracy	Precision	Recall	F_1–score
Adaboost	77.52%	0.7725	0.6803	0.7235
Random Forest	85.46%	**0.8960**	0.7509	0.8170
Bagging	**85.95%**	0.8894	0.7709	0.8259
K–NN	81.30%	0.8062	0.7471	0.7755
Linear SVM	78.97%	0.7789	0.7170	0.7467
Logistic regression	79.24%	0.7904	0.7073	0.7465
DNN	85.27%	0.8325	**0.8253**	**0.8289**
CNN	85.46%	0.8282	0.8109	0.8195

Table 7.7. Classification results for linguistic feature. Ensemble models outperform linear and deep models in symbolic representation space

Classifier	Accuracy	Precision	Recall	F_1–score
Adaboost	92.91%	0.9194	0.9163	0.9179
Random Forest	**95.63%**	0.9418	**0.9581**	0.9499
Bagging	95.29%	**0.9469**	0.9439	**0.9454**
Naïve Bayes	81.02%	0.7981	0.7509	0.7738
K–NN	84.82%	0.8261	0.8218	0.8239
Linear SVM	93.49%	0.9318	0.9166	0.9241
Logistic regression	93.40%	0.9295	0.9170	0.9232
DNN	94.94%	0.9331	0.9512	0.9421
CNN	94.86%	0.9414	0.9397	0.9406

Table 7.8. Classification results for hybrid feature. CNN model outperforms other models in terms of accuracy, precision, recall, and F_1-score

Classifier	Accuracy	Precision	Recall	F_1–score
Adaboost	92.76%	0.9143	0.9187	0.9165
Random Forest	96.03%	0.9472	0.9574	0.9522
Bagging	96.18%	0.9461	0.9482	0.9472
K–NN	84.83%	0.8266	0.8215	0.8240
Linear SVM	93.60%	0.9233	0.9291	0.9262
Logistic regression	93.78%	0.9277	0.9284	0.9281
DNN	96.24%	0.9436	0.9261	0.9348
CNN	**96.88%**	**0.9551**	**0.9623**	**0.9586**

Table 7.9. Results of the McNemar's statistical test on the feature sets for the value prediction task

Feature Pair	χ^2 Estimation	P
Hybrid ⊗ LW2V	220.332	<0.005
Hybrid ⊗ W2V	244.803	<0.005
Hybrid ⊗ GloVe	204.537	<0.005
Hybrid ⊗ Linguistic	4.762	<0.05

Table 7.10. Results of the McNemar's statistical test on the classifiers for the value prediction task

Classifier Pair	χ^2 Estimation	P
CNN ⊗ Adaboost	104.378	<0.005
CNN ⊗ Random Forest	5.625	<0.025
CNN ⊗ Bagging	4.000	<0.025
CNN ⊗ K–NN	623.546	<0.005
CNN ⊗ Linear SVM	45.803	<0.005
CNN ⊗ Logistic regression	97.714	<0.005
CNN ⊗ DNN	4.923	<0.05

By analyzing these results, we observe the following:

(1) Our proposed hybrid feature outperforms other features in terms of accuracy and F_1–score. This is because it benefits from both implicit semantic and syntactic information embedded within word vectors and explicit engineered features.

(2) The best result is achieved by training a deep CNN on the proposed hybrid feature. In addition to mentioned advantages of the hybrid feature, the ability of a CNN model to

learn local filters plays an important role in deciding discriminative regions of feature and hence achieving better results.

(3) Deep learning models outperform other models in terms of accuracy, recall, and F_1–score in almost all feature sets except for linguistic feature. This is because deep models generalize better when data is high–dimensional and continuous.

(4) Among linear and ensemble models, random forest and bagging classifiers achieve better results.

(5) Among word vector models, pre–trained GloVe model results in better accuracy, recall, and F_1–score whereas pre–trained Word2Vec model results in a better precision. Also, pre–trained models outperform the locally–trained model. This implies that the dataset is not big enough to directly train embeddings on it.

(6) Linguistic features that exploit corpus information (i.e., linguistic and hybrid features) outperform the universal features.

(7) The McNemar's statistical tests shown in Tables 7.9 and 7.10 suggest that the differences between different feature sets and the classifiers are statistically significant.

We also used an unsupervised attribute value extraction model introduced in [124] to compare its performance with supervised models. This model resulted in an accuracy of 57.32%, precision of 0.6123, recall of 0.4912, and F_1–score of 0.5451. These results suggest that supervised models significantly outperform the unsupervised model (improvement of 39.56% on accuracy).

Finally, Figure 7.5 illustrates a set of sample models scaled by online–learning. Default sizes of these models are shown in the upper row whereas same models after being automatically scaled along with their extracted attributes (L: length, H: height, W: weight, and V: velocity) are shown in the lower row.

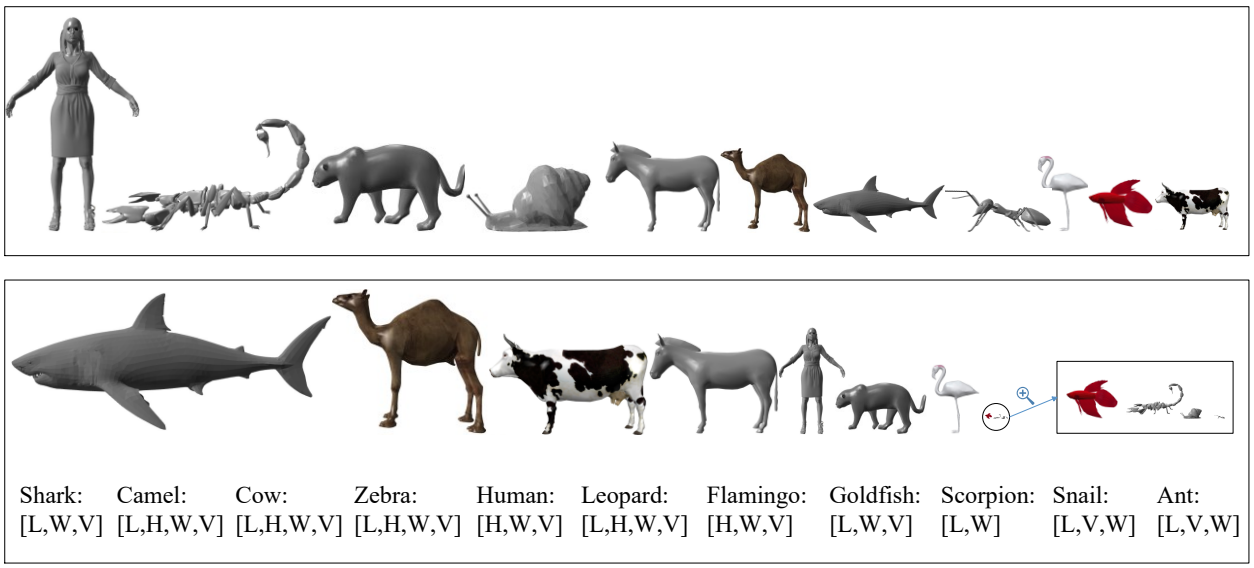

Figure 7.5. Online–learning to assign physical attributes to 3D models. The upper row shows a collection of 3D models in their default sizes and lower row shows same models after being automatically scaled in descending order in size. L: Length, H: Height, W: Weight, V: Velocity. For example, a cow is assigned of physical attribute values of length (2±0.4 m), height (1.6±0.2 m), weight (950±345 kg), and velocity (25±7.5 km/h)

We addressed predicting physical attributes and their corresponding values from the web in an online manner using ensemble and deep models. For this purpose, we used both structural information of the web documents and their content to produce rich feature sets. The results suggest that when combined with these feature sets, the deep models outperform other models in terms of attribute value prediction. Also, results show that a random forest classifier outperforms other linear and ensemble models on attribute prediction task.

Chapter 8. Conclusion

In this thesis, we proposed deep learning and probabilistic graphical models to address two general aspects of common–sense knowledge which are essential for automating 3D word modeling and animation and are specifically required by text–to–scene and text–to–animation conversion systems. These aspects include spatial relations between objects and their physical attributes. We showed that using supervised models can achieve excellent performance in these tasks.

8.1 Summary of Contributions

In chapter 4, we addressed two tasks: (1) coarse–grained word–sense disambiguation of spatial pivots, and (2) automatic spatial role labeling of a given utterance. We proposed a hybrid feature which is a combination of universal word embeddings and linguistic features of a corpus. We also designed a convolutional neural network which is trained on proposed hybrid feature while using its predictions from preceding words. We compiled two datasets and along with a dataset provided by SemEval performed comprehensive experiments using different learning models, and model–feature setups. Results suggested that our proposed approach significantly outperforms other methods in terms of F_1–score.

In chapter 5, we proposed a neural approach towards computational aspects of mapping spatial relations between a world model and a language model. This approach is inspired by a cognitive theory of spatial relations proposed in [9]. Results showed that combining geometric features with semantic features of objects improves both inferring and grounding tasks. Results also suggested that fully–connected feed–forward neural networks with three hidden layers outperform a variety of other learning models in both tasks. Moreover, we developed a probabilistic recommender system to facilitate developing virtual scenarios by providing a user with suggestions of relevant objects and their corresponding spatial arrangement. Subjective evaluations suggested that this module can recommend plausible object–relation pairs given an object of interest.

In chapter 6, we proposed an integrated approach towards learning physical attributes of objects using the web data. We used a semantic taxonomy merging approach to map object categories collected from heterogeneous resources to the WordNet hierarchy. We also used a logistic regression model to address missing categories from WordNet. We then trained an ensemble of Gaussian mixture models to model physical attributes from data collected from the web and dimension ratios extracted from a 3D model database. We used this ensemble model to enhance the learned distribution. We also proposed a Monte Carlo technique as an inference mechanism to infer comparative relations between physical attributes of objects. Results suggested that our proposed technique can automatically scale 3D models in soft real–time.

In chapter 7, we proposed an online learning approach to automatically extract physical attributes of a given object from the web. The proposed model consists of a random forest classifier trained to predict a set of physical attributes of an object, a deep convolutional neural network trained to predict values of a given object–attribute pair, and an unsupervised maximum likelihood estimator to estimate the parameters of value distributions. We also proposed a hybrid feature which simultaneously benefits from the explicit information of word embeddings and explicit linguistic knowledge. The proposed feature model combined with a convolutional neural network achieves excellent results on this task.

8.2 Future Work

8.2.1 Action Learning

The first direction is learning actions for articulated models. In most text–to–scene and text–to–animation conversion systems, actions are either limited to low–level translations and rotations, or to a small set of pre–planned actions. Hence, they do not address unrestricted and unseen actions expressed as a set of natural language descriptions. "*a horse is galloping*", "*a cat is chasing after a mouse*", "*a mother is stroking her daughter in bed*" are examples of such descriptions. To address this, we need to collect a dataset of annotated actions (extracted from videos using machine vision techniques or motion capturing sensors such as Kinect) and train models to retarget them on arbitrary 3D models.

8.2.2 Active Learning

Another possible direction is to exploit active learning (user–in–the–loop). Although we showed that supervised models achieve excellent results, yet they are limited to available data. As an example, our trained neural models for grounding spatial relations can position models in a 3D space based on 27 different spatial pivots. However, it is possible that an input utterance contains a spatial pivot that is not provided in training data. A reliable remedy for such scenarios is to let a system interact with users and update its knowledge from user interactions and feedbacks in an incremental manner. For instance, assume that a user asks our grounding neural model to "*Place a book beside a laptop*". This model positions a model of the laptop beside a model of the book. The user may then request to move the book closer to the laptop. This signal can be considered as a single training sample to update the neural model. Hence, the model can update its weights to place a book closer to a laptop in upcoming predictions.

8.2.3 Automatic Model Standardization

We addressed the automatic scaling of 3D models in chapters 6 and 7 to scale the models in a way that they show a consistent size with their physical counterparts. Another important standardization process on 3D models is to extract their correct *up* and *front* directions and then rotate them to match these directions. Given the huge number of object categories and variations among them, and also subjective nature of these directions, a feasible solution is to train a model on a collection of images of a query model to predict the most likely *up* and *front* directions of the query model.

References

[1] E. Davis, *Representations of Commonsense Knowledge*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990.

[2] E. Davis and G. Marcus, "Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence," *Commun ACM*, vol. 58, no. 9, pp. 92–103, Aug. 2015.

[3] R. Gupta and M. J. Kochenderfer, "Common Sense Data Acquisition for Indoor Mobile Robots," in *Proceedings of the 19th National Conference on Artifical Intelligence*, San Jose, California, 2004, pp. 605–610.

[4] P. Tarau and E. Figa, "Knowledge-based Conversational Agents and Virtual Storytelling," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, New York, NY, USA, 2004, pp. 39–44.

[5] B. Coyne and R. Sproat, "WordsEye: An Automatic Text-to-scene Conversion System," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2001, pp. 487–496.

[6] S. Antol *et al.*, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2425–2433.

[7] M. Ma, "Automatic Conversion of Natural Language to 3D Animation," doctoral, University of Ulster, Londonderry, 2006.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

[9] K. R. Coventry and S. C. Garrod, *Saying, Seeing and Acting: The Psychological Semantics of Spatial Prepositions*, 1 edition. Hove, East Sussex ; New York: Psychology Press, 2004.

[10] S. R. Clay and J. Wilhelms, "Put: language-based interactive manipulation of objects," *IEEE Comput. Graph. Appl.*, vol. 16, no. 2, pp. 31–39, Mar. 1996.

[11] D. Sánchez, "A methodology to learn ontological attributes from the Web," *Data Knowl. Eng.*, vol. 69, no. 6, pp. 573–597, 2010.

[12] D. Davidov and A. Rappoport, "Extraction and approximation of numerical attributes from the Web," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 1308–1317.

[13] M. Pasca, "Attribute Extraction from Synthetic Web Search Queries.," in *Proceedings of the 5th International Joint Conference on Natural Language Processing*, 2011, vol. 11, pp. 401–409.

[14] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2 edition. Upper Saddle River, N.J: Pearson, 2008.

[15] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks, "A closer look at skip-gram modelling," in *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, 2006, pp. 1–4.

[16] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A Neural Probabilistic Language Model," *J. Mach. Learn. Res.*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[17] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, vol. 14, pp. 1532–1543.
[18] D. Gildea and D. Jurafsky, "Automatic Labeling of Semantic Roles," *Comput Linguist*, vol. 28, no. 3, pp. 245–288, Sep. 2002.
[19] G. Miller, *WordNet: An Electronic Lexical Database*. Cambridge, Mass: A Bradford Book, 1998.
[20] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The Berkeley FrameNet Project," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 1998, pp. 86–90.
[21] H. Liu and P. Singh, "ConceptNet — A Practical Commonsense Reasoning Tool-Kit," *BT Technol. J.*, vol. 22, no. 4, pp. 211–226, Oct. 2004.
[22] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," 2016.
[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[24] L. Breiman, "Random Forests," *Mach Learn*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[25] L. Breiman, "Bagging Predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[26] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*, 1995, pp. 23–37.
[27] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA, 2001, pp. 282–289.
[28] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[29] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60.
[30] K. Toutanova and C. D. Manning, "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger," in *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, Stroudsburg, PA, USA, 2000, pp. 63–70.
[31] D. Klein and C. D. Manning, "Accurate Unlexicalized Parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 2003, pp. 423–430.
[32] K. Glass and S. Bangay, "Evaluating Parts-of-speech Taggers for Use in a Text-to-scene Conversion System," in *Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, Republic of South Africa, 2005, pp. 20–28.
[33] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a Large Annotated Corpus of English: The Penn Treebank," *Comput Linguist*, vol. 19, no. 2, pp. 313–330, Jun. 1993.

[34] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky, "Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules," *Comput. Linguist.*, vol. 39, no. 4, pp. 885–916, Jan. 2013.

[35] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Stroudsburg, PA, USA, 2005, pp. 363–370.

[36] Sebastian Schuster and Christopher D. Manning, "An Improved Representation for Natural Language Understanding Tasks," in *Proceedings of the 10th International Conference on Language Resources and Evaluation*, 2016.

[37] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Am. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, Sep. 1990.

[38] Z. S. Harris, "Distributional structure," *Word*, vol. 10, pp. 146–162, 1954.

[39] O. Levy and Y. Goldberg, "Neural Word Embedding as Implicit Matrix Factorization," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2177–2185.

[40] O. Levy, Y. Goldberg, and I. Dagan, "Improving Distributional Similarity with Lessons Learned from Word Embeddings," *Trans. Assoc. Comput. Linguist.*, vol. 3, pp. 211–225, 2015.

[41] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.

[42] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013, pp. 1–12.

[43] T. Luong, R. Socher, and C. Manning, "Better Word Representations with Recursive Neural Networks for Morphology," in *CoNLL*, 2013, pp. 104–113.

[44] O. Levy, Y. Goldberg, and I. Ramat-Gan, "Linguistic Regularities in Sparse and Explicit Word Representations," in *CoNLL*, 2014, pp. 171–180.

[45] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[46] D. Joshi, J. Z. Wang, and J. Li, "The Story Picturing Engine—a System for Automatic Text Illustration," *ACM Trans Multimed. Comput Commun Appl*, vol. 2, no. 1, pp. 68–89, Feb. 2006.

[47] X. Zhu, A. B. Goldberg, M. Eldawy, C. R. Dyer, and B. Strock, "A Text-to-picture Synthesis System for Augmenting Communication," in *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, Vancouver, British Columbia, Canada, 2007, pp. 1590–1595.

[48] D. Joshi, "The story picturing engine: finding elite images to illustrate a story using mutual reinforcement," in *In MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, 2004, pp. 119–126.

[49] R. Agrawal, S. Gollapudi, A. Kannan, and K. Kenthapadi, "Enriching Textbooks with Images," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2011, pp. 1847–1856.

[50] V. Srinivasarao and V. Varma, “Web Image Annotation Using an Effective Term Weighting,” in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed. Springer Berlin Heidelberg, 2012, pp. 286–296.
[51] Z. Chen, J. Hou, D. Zhang, and X. Qin, “An Annotation Rule Extraction Algorithm for Image Retrieval,” *Pattern Recogn Lett*, vol. 33, no. 10, pp. 1257–1268, Jul. 2012.
[52] C.-C. Chiang, “Interactive tool for image annotation using a semi-supervised and hierarchical approach,” *Comput. Stand. Interfaces*, vol. 35, no. 1, pp. 50–58, Jan. 2013.
[53] Z. Gong, L. H. U, and C. W. Cheang, “Web image indexing by using associated texts,” *Knowl. Inf. Syst.*, vol. 10, no. 2, pp. 243–264, Oct. 2005.
[54] D. Kılınç and A. Alpkocak, “An Expansion and Reranking Approach for Annotation-based Image Retrieval from Web,” *Expert Syst Appl*, vol. 38, no. 10, pp. 13121–13127, Sep. 2011.
[55] Y. Zhang, R. Jin, and Z.-H. Zhou, “Understanding bag-of-words model: a statistical framework,” *Int. J. Mach. Learn. Cybern.*, vol. 1, no. 1–4, pp. 43–52, Aug. 2010.
[56] J. Z. Wang, J. Li, and G. Wiederhold, “SIMPLIcity: semantics-sensitive integrated matching for picture libraries,” *Pattern Anal. Mach. Intell. IEEE Trans. On*, vol. 23, no. 9, pp. 947–963, Sep. 2001.
[57] R. Mihalcea and P. Tarau, “TextRank: Bringing order into texts,” in *Proceedings of Empirical Methods in Natural Language Processing*, 2004, pp. 404–411.
[58] G. Adorni, M. Manzo, and G. Ferrari, “Natural language input for scene generation,” in *Proceedings of the first conference on European chapter of the Association for Computational Linguistics*, 1983, pp. 175–182.
[59] M. Manzo, G. Adorni, and F. Giunchiglia, “Reasoning about scene descriptions,” *IEEE Proc. Nat. Lang.*, vol. 74, pp. 1013–1025, 1986.
[60] A. Herskovits, “Semantics and pragmatics of locative expressions,” *Cogn. Sci.*, vol. 9, no. 3, pp. 341–378, Jul. 1985.
[61] B. Coyne, O. Rambow, J. Hirschberg, and R. Sproat, “Frame Semantics in Text-to-Scene Generation,” in *Knowledge-Based and Intelligent Information and Engineering Systems*, R. Setchi, I. Jordanov, R. J. Howlett, and L. C. Jain, Eds. Springer Berlin Heidelberg, 2010, pp. 375–384.
[62] B. Coyne, R. Sproat, and J. Hirschberg, “Spatial relations in text-to-scene conversion,” in *Workshop at Spatial Cognition: Computational Models of Spatial Language Interpretation*, Mt. Hood, OR, USA, 2010, pp. 9–16.
[63] D. B. Bob Coyne, “VigNet: grounding language in graphics using frame semantics,” in *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, 2011, pp. 28–36.
[64] B. Coyne, A. Klapheke, M. Rouhizadeh, R. Sproat, and D. Bauer, “Annotation Tools and Knowledge Representation for a Text-To-Scene System,” in *Proceedings of COLING 2012*, 2012, pp. 679–694.
[65] K. Fort, G. Adda, and K. B. Cohen, “Amazon Mechanical Turk: Gold Mine or Coal Mine?,” *Comput. Linguist.*, vol. 37, no. 2, pp. 413–420, Apr. 2011.
[66] M. Rouhizadeh, B. Coyne, and R. Sproat, “Collecting Semantic Information for Locations in the Scenario-Based Lexical Knowledge Resource of a Text-to-Scene Conversion System,” in *Knowledge-Based and Intelligent Information and Engineering Systems*, A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, Eds. Springer Berlin Heidelberg, 2011, pp. 378–387.

[67] M. Rouhizadeh, M. Bowler, R. Sproat, and B. Coyne, “Collecting Semantic Data from Mechanical Turk for a Lexical Knowledge Resource in a Text to Picture Generating System,” in *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, 2011.

[68] M. Rouhizadeh, M. Bowler, R. Sproat, and B. Coyne, “Data collection and normalization for building the Scenario-Based Lexical Knowledge Resource of a text-to-scene conversion system,” in *2010 5th International Workshop on Semantic Media Adaptation and Personalization (SMAP)*, 2010, pp. 25–30.

[69] C. Spika, K. Schwarz, H. Dammertz, and H. P. A. Lensch, “AVDT - Automatic Visualization of Descriptive Texts,” in *Proceedings of the Vision, Modeling, and Visualization Workshop*, Berlin, German, 2011, pp. 129–136.

[70] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, “A framework and graphical development environment for robust NLP tools and applications,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA, USA, 2002, pp. 168–175.

[71] A. Chang, M. Savva, and C. Manning, “Interactive Learning of Spatial Knowledge for Text to 3D Scene Generation,” in *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 2014, pp. 14–21.

[72] A. Chang, M. Savva, and C. Manning, “Semantic Parsing for Text to 3D Scene Generation,” in *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, 2014, pp. 17–21.

[73] A. X. Chang, M. Savva, and C. D. Manning, “Learning Spatial Knowledge for Text to 3D Scene Generation.,” in *EMNLP*, 2014, pp. 2028–2038.

[74] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, “Example-based Synthesis of 3D Object Arrangements,” *ACM Trans Graph*, vol. 31, no. 6, p. 135:1–135:11, Nov. 2012.

[75] M. Savva, A. X. Chang, G. Bernstein, C. D. Manning, and P. Hanrahan, “On Being the Right Scale: Sizing Large Collections of 3D Models,” in *SIGGRAPH Asia 2014 Indoor Scene Understanding Where Graphics Meets Vision*, New York, NY, USA, 2014, p. 4:1–4:11.

[76] C. L. Zitnick, D. Parikh, and L. Vanderwende, “Learning the Visual Interpretation of Sentences,” in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, Washington, DC, USA, 2013, pp. 1681–1688.

[77] C. Quirk *et al.*, “MSR SPLAT, a Language Analysis Toolkit,” in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstration Session*, Stroudsburg, PA, USA, 2012, pp. 21–24.

[78] S. Chaudhuri, E. Kalogerakis, S. Giguere, and T. Funkhouser, “Attribit: Content Creation with Semantic Attributes,” in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 2013, pp. 193–202.

[79] T. Winograd, “Procedures as a Representation for Data in a Computer Program for Understanding Natural Language,” M.I.T. Artificial Intelligence Laboratory, Technical Report AITR-235, Jan. 1971.

[80] N. I. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, and M. Palmer, *Parameterized action representation for virtual human agents*. Cambridge, MA, USA: MIT Press, 2000.

[81] N. Badler, R. Bindiganavale, J. Bourne, M. Palmer, J. Shi, and W. Schuler, "A Parameterized Action Representation for Virtual Human Agents," in *EMBODIED CONVERSATIONAL AGENTS*, 1998, pp. 256–284.

[82] R. Bindiganavale, W. Schuler, J. M. Allbeck, N. I. Badler, A. K. Joshi, and M. Palmer, "Dynamically Altering Agent Behaviors Using Natural Language Instructions," in *Proceedings of the Fourth International Conference on Autonomous Agents*, New York, NY, USA, 2000, pp. 293–300.

[83] N. Badler, R. Bindiganavale, J. Bourne, M. Palmer, J. Shi, and W. Schuler, "A Parameterized Action Representation for Virtual Human Agents," in *Embodied Conversational Agents*, 2000, pp. 256–284.

[84] P. Paroubek, Y. Schabes, and A. K. Joshi, "XTAG: A Graphical Workbench for Developing Tree-adjoining Grammars," in *Proceedings of the Third Conference on Applied Natural Language Processing*, Stroudsburg, PA, USA, 1992, pp. 223–230.

[85] N. I. Badler, C. B. Phillips, and B. L. Webber, *Simulating Humans: Computer Graphics Animation and Control*. New York: Oxford University Press, 1993.

[86] S. Dupuy, A. Egges, V. Legendre, and P. Nugues, "Generating a 3d simulation of a car accident from a written description in natural language: The carsim system," in *Proceedings of the workshop on Temporal and spatial information processing*, 2001, vol. 13, pp. 1–8.

[87] O. Akerberg, H. Svensson, B. Schulz, and P. Nugues, "Carsim: an automatic 3d text-to-scene conversion system applied to road accident reports," in *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, 2003, vol. 2, pp. 191–194.

[88] R. Johansson, D. Williams, A. Berglund, and P. Nugues, "Carsim: a system to visualize written road accident reports as animated 3d scenes," in *Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, Association for Computational Linguistics, 2004, vol. 57–64.

[89] J. Carlberger and V. Kann, "Implementing an Efficient Part-of-speech Tagger," *Softw Pr. Exper*, vol. 29, no. 9, pp. 815–832, Jul. 1999.

[90] Z.-Q. Liu and K.-M. Leung, "Script visualization (ScriptViz): a smart system that makes writing fun," *Soft Comput.*, vol. 10, no. 1, pp. 34–40, Apr. 2005.

[91] S. Sekine, "Corpus-based parsing and sublanguage studies," Department of Computer Science, New York University, 1998.

[92] M. Ma and P. M. Kevitt, "Virtual human animation in natural language visualisation," *Artif. Intell. Rev.*, vol. 25, no. 1–2, pp. 37–53, Oct. 2007.

[93] T. Järvinen and P. Tapanainen, "A Dependency Parser for English," Department of General Linguistics, University of Helsinki, TR-1, 1997.

[94] M. Ma and P. M. Kevitt, "Visual Semantics and Ontology of Eventive Verbs," in *Natural Language Processing – IJCNLP 2004*, K.-Y. Su, J. Tsujii, J.-H. Lee, and O. Y. Kwong, Eds. Springer Berlin Heidelberg, 2005, pp. 187–196.

[95] E. Hanser, P. M. Kevitt, T. Lunney, J. Condell, and M. Ma, "SceneMaker: Multimodal Visualisation of Natural Language Film Scripts," in *Knowledge-Based and Intelligent Information and Engineering Systems*, R. Setchi, I. Jordanov, R. J. Howlett, and L. C. Jain, Eds. Springer Berlin Heidelberg, 2010, pp. 430–439.

[96] E. Hanser, P. M. Kevitt, T. Lunney, and J. Condell, "SceneMaker: Automatic Visualisation of Screenplays," in *KI 2009: Advances in Artificial Intelligence*, B. Mertsching, M. Hund, and Z. Aziz, Eds. Springer Berlin Heidelberg, 2009, pp. 265–272.

[97] R. Valitutti, “WordNet-Affect: an Affective Extension of WordNet,” in *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, 2004, pp. 1083–1086.
[98] M. Oshita, “Generating Animation from Natural Language Texts and Framework of Motion Database,” in *International Conference on CyberWorlds, 2009. CW ’09*, 2009, pp. 146–153.
[99] M. Oshita, “Generating animation from natural language texts and semantic analysis for motion search and scheduling,” *Vis. Comput.*, vol. 26, no. 5, pp. 339–352, Feb. 2010.
[100] C. Fillmore, “The Case for Case,” in *Universals in Linguistic Theory*, New York: Holt, Rinehart, 1968, pp. 1–88.
[101] K. Hassani, A. Nahvi, and A. Ahmadi, “Architectural design and implementation of intelligent embodied conversational agents using fuzzy knowledge base,” *J. Intell. Fuzzy Syst.*, vol. 25, no. 3, pp. 811–823, Jan. 2013.
[102] K. Hassani, A. Nahvi, and A. Ahmadi, “Design and implementation of an intelligent virtual environment for improving speaking and listening skills,” *Interact. Learn. Environ.*, vol. 0, pp. 1–20, Oct. 2013.
[103] Y. Takashima, H. Shimazu, and M. Tomono, “Story Driven Animation,” in *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*, New York, NY, USA, 1987, pp. 149–153.
[104] X. Zeng, Q. Mehdi, and N. Gough, “3D Scene Creation Using Story-Based Descriptions,” in *Proceedings of CGAIMS’2005*, Louisville, Kentucky, USA, 2005, pp. 74–80.
[105] X. Zeng, Q. H. Mehdi, and N. E. Gough, “From visual semantic parameterization to graphic visualization,” in *Ninth International Conference on Information Visualisation, 2005. Proceedings*, 2005, pp. 488–493.
[106] X. Zeng and M. Tan, “The Development of a Language Interface for 3D Scene Generation,” in *Proceedings of the Second IASTED International Conference on Human Computer Interaction*, Anaheim, CA, USA, 2007, pp. 136–141.
[107] X. Zeng, “Generation of a 3D virtual environment based on story descriptions,” PhD Dissertation, University of Wolverhampton, 2007.
[108] K. Sumi and K. Tanaka, “Automatic Conversion from E-Content into Virtual Storytelling,” in *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*, G. Subsol, Ed. Springer Berlin Heidelberg, 2005, pp. 260–269.
[109] K. Sumi and M. Nagata, “Animated Storytelling System via Text,” in *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, New York, NY, USA, 2006.
[110] K. Glass, “Automating the Conversion of Natural Language Fiction to Multi-Modal 3D Animated Virtual Environments,” PhD Dissertation, Department of Computer Science, Rhodes University, 2008.
[111] K. Glass and S. Bangay, “Automating the creation of 3D animation from annotated fiction text,” in *Proceedings of the IADIS International conference on Computer Graphics and Visualization*, 2008, pp. 3–10.
[112] P. Ye and T. Baldwin, “Towards Automatic Animated Storyboarding,” in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, Chicago, Illinois, 2008, pp. 578–583.

[113] A. Ratnaparkhi, “A Maximum Entropy Model for Part-Of-Speech Tagging,” 1996, pp. 133–142.
[114] H. Shim, B. Kang, and K. Kwag, “Web2Animation - Automatic Generation of 3D Animation from the Web Text,” in *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT ’09*, 2009, vol. 1, pp. 596–601.
[115] W. Ward, “Understanding spontaneous speech: the Phoenix system,” in *, 1991 International Conference on Acoustics, Speech, and Signal Processing, 1991. ICASSP-91*, 1991, pp. 365–367 vol.1.
[116] A. Oddie, P. Hazlewood, B. Farrimond, and S. Presland, “Applying Deductive Techniques to the Creation of Realistic Historical 3D Spatiotemporal Visualisations from Natural Language Narratives,” in *Proceedings of the 2011 International Conference on Electronic Visualisation and the Arts*, Swinton, UK, UK, 2011, pp. 97–105.
[117] S. Presland, B. Farrimond, P. Hazlewood, and A. Oddie, “Creating Complex Interactive 3D Visualisations of Naval Battles from Natural Language Narratives,” in *Developments in E-systems Engineering (DESE), 2010*, 2010, pp. 113–118.
[118] E. Bolaño-Rodríguez, J. C. González-Moreno, D. Ramos-Valcarcel, and L. Vázquez-López, “Using Multi-Agent Systems to Visualize Text Descriptions,” in *Advances on Practical Applications of Agents and Multiagent Systems*, Y. Demazeau, M. Pěchoucěk, J. M. Corchado, and J. B. Pérez, Eds. Springer Berlin Heidelberg, 2011, pp. 39–45.
[119] J. C. G. Moreno and L. V. López, “Using Techniques Based on Natural Language in the Development Process of Multiagent Systems,” in *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, J. M. Corchado, S. Rodríguez, J. Llinas, and J. M. Molina, Eds. Springer Berlin Heidelberg, 2009, pp. 269–273.
[120] J. Pavón and J. Gómez-Sanz, “Agent Oriented Software Engineering with INGENIAS,” in *Multi-Agent Systems and Applications III*, V. Mařík, M. Pěchouček, and J. Müller, Eds. Springer Berlin Heidelberg, 2003, pp. 394–403.
[121] C. Kelleher and R. Pausch, “Using Storytelling to Motivate Programming,” *Commun ACM*, vol. 50, no. 7, pp. 58–64, Jul. 2007.
[122] S. Lee and J. Yan, “The Potential of a Text-Based Interface as a Design Medium: An Experiment in a Computer Animation Environment,” *Interact. Comput.*, Sep. 2014.
[123] G. Adorni, M. Manzo, and F. Giunchiglia, “Natural language driven image generation,” in *Proceedings of COLING 84*, 1984, pp. 495–500.
[124] K. Hassani and W.-S. Lee, “Adaptive animation generation using web content mining,” in *2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2015, pp. 1–8.
[125] X. Zeng, Q. Mehdi, and N. Gough, “3D Scene Creation Using Story-Based Descriptions,” in *Proceedings of CGAIMS’2005*, Louisville, Kentucky, USA, 2005, pp. 74–80.
[126] B. Landau and R. Jackendoff, “‘What’ and ‘where’ in spatial language and spatial cognition,” *Behav. Brain Sci.*, vol. 16, no. 02, pp. 217–238, Jun. 1993.
[127] K. Litkowski and O. Hargraves, “SemEval-2007 Task 06: Word-sense Disambiguation of Prepositions,” in *Proceedings of the 4th International Workshop on Semantic Evaluations*, Stroudsburg, PA, USA, 2007, pp. 24–29.
[128] S. Tratz and D. Hovy, “Disambiguation of Preposition Sense Using Linguistically Motivated Features,” in *Proceedings of Human Language Technologies: The 2009 Annual*

Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium, Stroudsburg, PA, USA, 2009, pp. 96–100.

[129] A. Herskovits, "Language, Spatial Cognition, and Vision," in *Spatial and Temporal Reasoning*, O. Stock, Ed. Springer Netherlands, 1997, pp. 155–202.

[130] J. Xu and D. M. Mark, "Natural Language Understanding of Spatial Relations Between Linear Geographic Objects," *Spat. Cogn. Comput.*, vol. 7, no. 4, pp. 311–347, Dec. 2007.

[131] P. Kordjamshidi, M. Van Otterlo, and M.-F. Moens, "Spatial Role Labeling: Towards Extraction of Spatial Relations from Natural Language," *ACM Trans Speech Lang Process*, vol. 8, no. 3, p. 4:1–4:36, Dec. 2011.

[132] R. Navigli, "Word Sense Disambiguation: A Survey," *ACM Comput Surv*, vol. 41, no. 2, p. 10:1–10:69, Feb. 2009.

[133] A. Dittrich, M. Vasardani, S. Winter, T. Baldwin, and F. Liu, "A Classification Schema for Fast Disambiguation of Spatial Prepositions," in *Proceedings of the 6th ACM SIGSPATIAL International Workshop on GeoStreaming*, New York, NY, USA, 2015, pp. 78–86.

[134] Kirk Roberts, Michael A. Skinner, and Sanda M. Harabagiu, "Recognizing Spatial Containment Relations between Event Mentions," presented at the 10th International Conference on Computational Semantics, 2013.

[135] J. Fasola and M. J. Matarić, "Using Spatial Semantic and Pragmatic Fields to Interpret Natural Language Pick-and-Place Instructions for a Mobile Service Robot," in *Proceedings of the 5th International Conference on Social Robotics - Volume 8239*, New York, NY, USA, 2013, pp. 501–510.

[136] J. Fasola and M. J. Matarić, "Interpreting instruction sequences in spatial language discourse with pragmatics towards natural human-robot interaction," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2720–2727.

[137] J. Fasola and M. J. Mataric, "Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 143–150.

[138] Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, Luca Iocchi, Roberto Basili, and Daniele Nardi, "HuRIC: a Human Robot Interaction Corpus," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014, pp. 4519–4526.

[139] K. Dukes, "Semeval-2014 task 6: Supervised semantic parsing of robotic spatial commands.," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, 2014, pp. 45–53.

[140] M. Skubic *et al.*, "Spatial Language for Human-robot Dialogs," *Trans Sys Man Cyber Part C*, vol. 34, no. 2, pp. 154–167, May 2004.

[141] M. Bhatt, J. H. Lee, and C. Schultz, "CLP(QS): A Declarative Spatial Reasoning Framework," in *Spatial Information Theory*, M. Egenhofer, N. Giudice, R. Moratz, and M. Worboys, Eds. Springer Berlin Heidelberg, 2011, pp. 210–230.

[142] J. L. Leidner, "Towards a reference corpus for automatic toponym resolution evaluation," in *Workshop on Geographic Information Retrieval*, Sheffield, UK, 2004.

[143] R. Q. Inderjeet Mani Janet Hitzeman, Justin Richer, Dave Harris and B. Wellner, "SpatialML: Annotation Scheme, Corpora, and Tools," in *Proceedings of the Sixth*

International Conference on Language Resources and Evaluation (LREC'08), Marrakech, Morocco, 28-30.

[144] J. Pustejovsky, J. Moszkowicz, and M. Verhagen, "Iso-space: The annotation of spatial information in language," in *Proceedings of the Sixth Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, 2011, pp. 1–9.

[145] Jordan Zlatev, "Spatial semantics," in *The Oxford Handbook of Cognitive Linguistics*, Dirk Geeraerts and Hubert Cuyckens, Eds. Oxford University Press, 2007, pp. 318–350.

[146] O. Ludwig, X. Liu, P. Kordjamshidi, and M.-F. Moens, "Deep Embedding for Spatial Role Labeling," *ArXiv160308474 Cs*, Mar. 2016.

[147] P. Kordjamshidi, S. Bethard, and M.-F. Moens, "SemEval-2012 Task 3: Spatial Role Labeling," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, Stroudsburg, PA, USA, 2012, pp. 365–373.

[148] O. Kolomiyets, P. Kordjamshidi, S. Bethard, and M.-F. Moens, "Semeval-2013 task 3: Spatial role labeling," in *Second joint conference on lexical and computational semantics (* SEM), Volume 2: Proceedings of the seventh international workshop on semantic evaluation (SemEval 2013)*, 2013, pp. 255–266.

[149] K. Roberts and S. M. Harabagiu, "UTD-SpRL: A Joint Approach to Spatial Role Labeling," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, Stroudsburg, PA, USA, 2012, pp. 419–424.

[150] Emanuele Bastianelli, Danilo Croce, Daniele Nardi, and Roberto Basili, "UNITOR-HMM-TK: Structured Kernel-based Learning for Spatial Role Labeling," in *Second Joint Conference on Lexical and Computational Semantics, Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, Atlanta, Georgia, 2013, pp. 573–579.

[151] J. A. Bateman, J. Hois, R. Ross, and T. Tenbrink, "A linguistic ontology of space for natural language processing," *Artif. Intell.*, vol. 174, no. 14, pp. 1027–1071, Sep. 2010.

[152] J. A. Bateman, B. Magnini, and G. Fabris, "The generalized upper model knowledge base: Organization and use," *Very Large Knowl. Bases*, pp. 60–72, 1995.

[153] P. Kordjamshidi and M.-F. Moens, "Global machine learning for spatial ontology population," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 30, pp. 3–21, Jan. 2015.

[154] Eduardo Blanco and Alakananda Vempala, "Inferring Temporally-Anchored Spatial Knowledge from Semantic Roles," in *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL,* Denver, Colorado, 2015, pp. 452–461.

[155] Alakananda Vempala and Eduardo Blanco, "Beyond Plain Spatial Knowledge: Determining Where Entities Are and Are Not Located, and For How Long," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 1502–1512.

[156] J. Yu, C. Li, W. Hong, S. Li, and D. Mei, "A new approach of rules extraction for word sense disambiguation by features of attributes," *Appl. Soft Comput.*, vol. 27, pp. 411–419, Feb. 2015.

[157] T. O'Hara and J. Wiebe, "Exploiting Semantic Role Resources for Preposition Disambiguation," *Comput Linguist*, vol. 35, no. 2, pp. 151–184, Jun. 2009.

[158] D. B. Lenat and R. V. Guha, *Building Large –d Systems; Representation and Inference in the Cyc Project*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
[159] D. Dahlmeier, H. T. Ng, and T. Schultz, "Joint learning of preposition senses and semantic roles of prepositional phrases," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 2009, pp. 450–458.
[160] P. Ye and T. Baldwin, "MELB-YB: Preposition Sense Disambiguation Using Rich Semantic Features," in *Proceedings of the 4th International Workshop on Semantic Evaluations*, Stroudsburg, PA, USA, 2007, pp. 241–244.
[161] D. Hovy, S. Tratz, and E. Hovy, "What's in a Preposition?: Dimensions of Sense Disambiguation for an Interesting Word Class," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Stroudsburg, PA, USA, 2010, pp. 454–462.
[162] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, pp. 160–167.
[163] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 2002, pp. 63–70.
[164] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, pp. 45–50.
[165] K. Litkowski, "Pattern Dictionary of English Prepositions," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 1274–1283.
[166] M. Grubinger, P. D. Clough, H. Müller, and T. Deselaers, "The IAPR Benchmark: A New Evaluation Resource for Visual Information Systems," presented at the International Conference on Language Resources and Evaluation, Genoa, Italy, 2006.
[167] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J Mach Learn Res*, vol. 12, pp. 2825–2830, Nov. 2011.
[168] Martín Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015.
[169] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015, pp. 1–15.
[170] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[171] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," presented at the Proceedings of The 32nd International Conference on Machine Learning, 2015, pp. 448–456.
[172] A. K. McCallum, "MALLET: A Machine Learning for Language Toolkit," 2002.
[173] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds. Springer New York, 1990, pp. 167–193.

[174] C. Hudelot, J. Atif, and I. Bloch, “Fuzzy spatial relation ontology for image interpretation,” *Fuzzy Sets Syst.*, vol. 159, no. 15, pp. 1929–1951, Aug. 2008.
[175] M. J. EGENHOFER and R. D. FRANZOSA, “Point-set topological spatial relations,” *Int. J. Geogr. Inf. Syst.*, vol. 5, no. 2, pp. 161–174, Jan. 1991.
[176] G. D. Logan and D. D. Sadler, “A computational analysis of the apprehension of spatial relations,” in *Language and space*, P. Bloom, M. A. Peterson, L. Nadel, and M. F. Garrett, Eds. Cambridge, MA, US: The MIT Press, 1996, pp. 493–529.
[177] T. Regier and L. A. Carlson, “Grounding spatial language in perception: an empirical and computational investigation,” *J. Exp. Psychol. Gen.*, vol. 130, no. 2, pp. 273–298, Jun. 2001.
[178] S. Schockaert, C. Cornelis, M. D. Cock, and E. E. Kerre, “Fuzzy Spatial Relations between Vague Regions,” in *2006 3rd International IEEE Conference Intelligent Systems*, 2006, pp. 221–226.
[179] J. M. Keller and X. Wang, “A Fuzzy Rule-Based Approach to Scene Description Involving Spatial Relationships,” *Comput Vis Image Underst*, vol. 80, no. 1, pp. 21–41, Oct. 2000.
[180] Xiaomei Wang and J. M. Keller, “Human-based spatial relationship generalization through neural/fuzzy approaches,” *Fuzzy Sets Syst.*, vol. 101, no. 1, pp. 5–20, Jan. 1999.
[181] R. Frances Wang, “Action, verbal response and spatial reasoning,” *Cognition*, vol. 94, no. 2, pp. 185–192, Dec. 2004.
[182] L. A. Carlson, R. West, H. A. Taylor, and R. W. Herndon, “Neural correlates of spatial term use,” *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 28, no. 6, pp. 1391–1408, 2002.
[183] W. G. Hayward and M. J. Tarr, “Spatial language and spatial representation,” *Cognition*, vol. 55, no. 1, pp. 39–84, Apr. 1995.
[184] X. Wang, P. Matsakis, L. Trick, B. Nonnecke, and M. Veltman, “A Study on how Humans Describe Relative Positions of Image Objects,” in *Headway in Spatial Data Handling*, A. Ruas and C. Gold, Eds. Springer Berlin Heidelberg, 2008, pp. 1–18.
[185] L. E. Crawford, T. Regier, and J. Huttenlocher, “Linguistic and non-linguistic spatial categorization,” *Cognition*, vol. 75, no. 3, pp. 209–235, Jun. 2000.
[186] E. Munnich, B. Landau, and B. A. Dosher, “Spatial language and spatial representation: a cross-linguistic comparison,” *Cognition*, vol. 81, no. 3, pp. 171–207, Oct. 2001.
[187] P. Li and L. Gleitman, “Turning the tables: language and spatial reasoning,” *Cognition*, vol. 83, no. 3, pp. 265–294, Apr. 2002.
[188] R. F. Wang, “Spatial Representations and Spatial Updating,” vol. 42, pp. 109–156, 2003.
[189] T. Blickle and L. Thiele, “A Comparison of Selection Schemes Used in Evolutionary Algorithms,” *Evol Comput*, vol. 4, no. 4, pp. 361–394, Dec. 1996.
[190] A. X. Chang *et al.*, “Shapenet: An information-rich 3d model repository,” *ArXiv Prepr. ArXiv151203012*, 2015.
[191] M. Savva, A. X. Chang, and P. Hanrahan, “Semantically-Enriched 3D Models for Common-sense Knowledge,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 24–31.
[192] H. Bagherinezhad, H. Hajishirzi, Y. Choi, and A. Farhadi, “Are Elephants Bigger than Butterflies? Reasoning about Sizes of Objects,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 2016, pp. 3449–3456.

[193] N. Tandon, G. De Melo, and G. Weikum, "Acquiring Comparative Commonsense Knowledge from the Web.," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 166–172.
[194] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.
[195] C. Corley and R. Mihalcea, "Measuring the semantic similarity of texts," in *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, 2005, pp. 13–18.
[196] D. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*. New York, NY, USA: Cambridge University Press, 2005.
[197] T. Lee, Z. Wang, H. Wang, and S. Hwang, "Attribute extraction and scoring: A probabilistic approach," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, 2013, pp. 194–205.
[198] X. Zheng, Y. Gu, and Y. Li, "Data extraction from web pages based on structural-semantic entropy," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 93–102.
[199] H. Takamura and J. Tsujii, "Estimating numerical attributes by bringing together fragmentary clues," in *The 2015 Annual Conference of the North American Chapter of the ACL*, 2015, pp. 1305–1310.
[200] A. Talaika, J. Biega, A. Amarilli, and F. M. Suchanek, "IBEX: Harvesting Entities from the Web Using Unique Identifiers," in *Proceedings of the 18th International Workshop on Web and Databases*, 2015, pp. 13–19.
[201] A. S. Maiya, D. Visser, and A. Wan, "Mining Measured Information from Text," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 899–902.
[202] A. Bakalov, A. Fuxman, P. P. Talukdar, and S. Chakrabarti, "Scad: Collective discovery of attribute values," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 447–456.

Appendix A: PENN Treebank Tag Set

Word Level			
CC	Coordinating conjunction	*CD*	Cardinal number
DT	Determiner	*EX*	Existential there
FW	Foreign word	*IN*	Prep. or subordinating conj.
JJ	Adjective	*JJR*	Adjective, comparative
JJS	Adjective, superlative	*LS*	List item marker
MD	Modal	*NN*	Noun, singular or mass
NNS	Noun, plural	*NNP*	Proper noun, singular
NNPS	Proper noun, plural	*PDT*	Pre-determiner
POS	Possessive ending	*PRP*	Personal pronoun
PRP$	Possessive pronoun	*RB*	Adverb
RBR	Adverb, comparative	*RBS*	Adverb, superlative
RP	Particle	*SYM*	Symbol
TO	to	*UH*	Interjection
VB	Verb, base form	*VBD*	Verb, past tense
VBG	Verb, gerund or present participle	*VBN*	Verb, past participle
VBP	Verb, non-3rd person singular present	*VBZ*	Verb, 3rd person singular present
WDT	Wh-determiner	*WP*	Wh-pronoun
WP$	Possessive wh-pronoun	*WRB*	Wh-adverb
Phrase Level			
ADJP	Adjective Phrase	*ADVP*	Adverb Phrase
CONJP	Conjunction Phrase	*FRAG*	Fragment
INTJ	Interjection	*LST*	List marker
NAC	Not a Constituent	*NP*	Noun Phrase
NX	Head of the NP in complex NPs	*PP*	Prepositional Phrase
PRN	Parenthetical	*PRT*	Particle
QP	Quantifier Phrase	*RRC*	Reduced Relative Clause
UCP	Unlike Coordinated Phrase	*VP*	Verb Phrase
WHADJP	Wh-adjective Phrase	*WHNP*	Wh-noun Phrase
WHADVP	Wh-adverb Phrase	*WHPP*	Wh-prepositional Phrase
Clause Level			
S	Simple declarative clause	*SINV*	Inverted declarative sentence
SBAR	Clause by a subordinating conjunction	*SQ*	Inverted yes/no question
SBARQ	Direct question by a wh-phrase		

Appendix B: Universal Dependency Tag Set

acl	clausal modifier of noun	*advcl*	adverbial clause modifier
advmod	adverbial modifier	*amod*	adjectival modifier
appos	appositional modifier	*aux*	auxiliary
auxpass	passive auxiliary	*case*	case marking
cc	coordination	*ccomp*	clausal complement
compound	compound	*conj*	conjunct
cop	copula	*csubj*	clausal subject
csubjpass	clausal passive subject	*dep*	dependent
det	determiner	*discourse*	discourse element
dislocated	dislocated elements	*dobj*	direct object
expl	expletive	*foreign*	foreign words
goeswith	goes with	*iobj*	indirect object
list	list	*mark*	marker
mwe	multi-word expression	*name*	name
neg	negation modifier	*nmod*	nominal modifier
nsubj	nominal subject	*nsubjpass*	passive nominal subject
nummod	numeric modifier	*parataxis*	parataxis
punct	punctuation	*remnant*	remnant in ellipsis
reparandum	overridden disfluency	*root*	root
vocative	vocative	*xcomp*	open clausal complement

Appendix C: Hyper–parameters for Disambiguating Spatial Preposition

Model	Hyper–parameters
Random Forest	*n_estimators* → 26 *max_depth* → 29 *min_samples_split* → 19 *min_samples_leaf* → 8 *bootstrap* → True *criterion* → entropy
AdaBoost	*n_estimators* → 23 *learning_rate* → 0.98
Bagging	*n_estimators* → 16 *bootstrap* → True *bootstrap_features* → False
Linear SVM	*C* → 1.06 *loss* → hinge *penalty* → l2 *tol* → 0.0012 *fit_intercept* → True *intercept_scaling* → 1.14
K–NN	*alpha* → 0.78 *n_neighbors* → 25 *weights* → distance *leaf_size* → 37 *metric* → minkowski
Naïve Bayes	*alpha* → 0.96 *fit_prior* → True
Logistic Regression	*penalty* → l2 *C* → 1.19 *fit_intercept* → True

*For descriptions of the hyper–parameters refer to http://scikit-learn.org

Appendix D: Hyper–parameters for Inferring Spatial Relations

Model	Hyper–parameters
Random Forest	*n_estimators* → 19 *max_depth* → 25 *min_samples_split* → 12 *min_samples_leaf* → 11 *bootstrap* → True *criterion* → gini
AdaBoost	*n_estimators* → 28 *learning_rate* → 1.01
Bagging	*n_estimators* → 21 *bootstrap* → True *bootstrap_features* → False
Logistic Regression	*penalty* → l2 *C* → 1.06 *fit_intercept* → True
RBF SVM	*C* → 1.0 *gamma* → 0.10 *degree* → 4 *kernel* → rbf
K–NN	*alpha* → 0.96 *n_neighbors* → 19 *weights* → unifrom *leaf_size* → 31 *metric* → minkowski

*For descriptions of the hyper–parameters refer to http://scikit-learn.org

Appendix E: Hyper–parameters for Grounding Spatial Relations

Regression Model	Hyper–parameters
Random Forest	*bootstrap* → True *criterion* → mse *min_samples_leaf* → 1 *min_samples_split* → 2 *n_estimators* → 26
AdaBoost	*learning_rate* → 1.02 *loss* → linear *n_estimators* → 27
Bagging	*bootstrap* → True *bootstrap_features* → False *n_estimators* → 26
Linear	*fit_intercept* → True
ARD	*alpha_1* → 0.0005 *alpha_2* → 0.0002 *fit_intercept* → True *lambda_1* → 0.0001 *lambda_2* → 0.0001
ElasticNet	*alpha* → 1.12 *l1_ratio* → 0.7
Lasso	*alpha* → 0.98
Lars	*eps* → 2.22e-16
SVR	*C* → 1.0 *degree* → 3 *epsilon* → 0.07 *kernel* → rbf
K–NN	*leaf_size* → 16 *metric* → minkowski *n_neighbors* → 16 *weights* → uniform

*For descriptions of the hyper–parameters refer to http://scikit-learn.org

Appendix F: Hyper–parameters for Online-Learning Physical Attributes

Model	Hyper–parameters
Random Forest	*n_estimators* → 17 *max_depth* → 19 *min_samples_split* → 18 *min_samples_leaf* → 15 *bootstrap* → True *criterion* → gini
AdaBoost	*n_estimators* → 26 *learning_rate* → 1.00
Bagging	*n_estimators* → 30 *bootstrap* → True *bootstrap_features* → False
Logistic Regression	*penalty* → l2 *C* → 1.00 *fit_intercept* → True
Linear SVM	*C* → 1.00 *loss* → hinge *penalty* → l2 *tol* → 0.0001 *fit_intercept* → True *intercept_scaling* → 1.01
K–NN	*alpha* → 1.00 *n_neighbors* → 30 *weights* → unifrom *leaf_size* → 25 *metric* → minkowski

*For descriptions of the hyper–parameters refer to http://scikit-learn.org

Appendix G: An example of hybrid features

Sentence: *The book is here*.

Unigrams: [the] [book] [is] [here] [.]

Lemmas: [the] [book] [be] [here] [.]

1–skip–bigrams: [the book] [the is] [book is] [book here] [is here] [is .] [here .]

Probability of unigrams belonging to spatial class: [0.50] [0.75] [0.50] [0.95] [0.50]

Probability of 1–skip–bigrams belonging to spatial class:

[0.75] [0.40] [0.65] [0.90] [0.85] [0.35] [0.75]

Word–level POS–tags: [DT] [NN] [VBZ] [RB] [.]

Phrase–level POS–tags of words: [NP] [NP] [VP] [ADVP] [.]

Phrase–level POS–tags of immediate ancestors of words: [S] [S] [S] [S] [S]

Minimum syntactic distance between words in a context window and a center word:

[2] [2] [0] [2] [2]

Collapsed dependencies between a center word and words within a context window:

[root (ROOT, is)] [nsubj (is, book)] [advmod (is, here)]

Named–entities: [O] [O] [O] [O] [O]

An indicator to decide whether a word refers to an abstract or a physical entity:

[0] [1] [0] [0] [0]

Polysemy score of words:

[-1] [0.091] [0.077] [0.5] [-1]

Word vectors:

The: [-0.172852 0.279297 0.106934 -0.158203 -0.084473 0.059082 0.040771 0.002548 0.259766 0.180664 0.097656 -0.081055 -0.010498 0.098145 0.000603 0.070801 -0.015625 -

0.095215 -0.081055 -0.028687 -0.033203 0.165039 0.039795 -0.037109 0.041016 -0.126953 -0.128906 0.123535 0.049805 0.012573 0.057861 -0.008301 -0.028320 -0.033203 0.161133 0.075195 -0.259766 0.089355 0.135742 0.004608 -0.044189 0.023193 -0.104492 -0.051514 0.083496 -0.020508 -0.021729 -0.027344 0.160156 0.190430 -0.032471 0.067871 0.103027 -0.253906 0.006348 0.205078 0.021118 -0.216797 -0.024414 0.170898 -0.218750 0.100098 -0.155273 -0.125977 -0.038330 -0.054199 0.192383 0.217773 0.121094 -0.026489 0.052979 -0.020142 0.053467 0.076660 0.045654 0.019775 0.124512 0.102051 0.152344 0.251953 0.042969 -0.185547 -0.075195 0.227539 -0.131836 -0.275391 -0.201172 0.125977 -0.186523 -0.139648 -0.125977 0.080078 -0.033936 -0.237305 -0.007660 0.283203 -0.017212 0.027222 -0.234375 0.128906 0.167969 -0.010193 0.065430 -0.132812 0.077637 -0.116699 0.014648 -0.052979 -0.121094 -0.099121 0.055908 -0.041992 0.107422 0.091309 0.147461 0.102051 -0.001396 0.201172 0.084473 0.075684 -0.243164 -0.189453 -0.079102 -0.066406 0.127930 0.150391 -0.166016 -0.095215 0.096191 -0.050293 -0.025024 -0.049316 -0.043701 -0.023438 0.066406 -0.133789 -0.250000 -0.242188 -0.113281 0.026367 0.138672 -0.070801 -0.222656 -0.112793 -0.018066 -0.190430 0.074219 -0.166016 0.006287 0.195312 -0.004364 -0.166992 0.147461 -0.048828 0.187500 -0.234375 0.144531 0.007690 0.041016 -0.067383 -0.016724 -0.050293 0.084961 0.036621 0.134766 0.080566 -0.044678 0.166016 -0.222656 0.009338 -0.119629 -0.064941 0.033447 -0.097656 0.014648 0.121582 0.123535 -0.134766 -0.052979 -0.277344 -0.037842 -0.079102 -0.068848 0.057129 0.031128 0.070801 0.079590 0.006561 0.142578 -0.084473 0.031250 -0.035645 0.023071 -0.016602 -0.172852 -0.161133 -0.107422 0.055664 -0.092285 -0.255859 0.039307 -0.204102 0.084961 0.046143 0.012085 -0.012573 0.063965 0.257812 0.184570 -0.055908 0.023438 0.104980 0.120117 0.009094 -0.005676 0.111328 0.151367 0.024658 -0.038330 0.115723 0.171875 -0.041260 0.103027 0.044678 -0.012695 -0.082031 -0.034912 0.007874 0.079102 -0.000504 0.022461 -0.161133 0.116211 0.251953 -0.046631 -0.080078 -0.098633 0.047607 -0.060547 0.118652 -0.033936 -0.066895 -0.000778 -0.048828 -0.011902 -0.082520 0.217773 -0.001808 0.148438 -0.020508 0.263672 0.170898 -0.046387 0.127930 0.048096 -0.037598 0.166992 -0.191406 0.039062 0.077148 -0.015869 0.099121 -0.158203 -0.013794 0.018555 0.155273 -0.181641 0.033691 -0.000805 0.066895 -0.018433 0.141602 0.064453 -0.007324 0.079590 -0.027954 0.125000 -0.237305 0.036377 -0.046875 -0.046875 0.054932 0.079590 0.271484 0.225586 -0.205078 -0.205078 -0.064453 -0.077637 -0.069824 -0.017700 -0.128906 0.021973 0.014771 -0.052979 -0.203125 0.061768 0.123047 0.129883 -0.182617]

desk: [-0.038086 0.025146 -0.341797 0.051270 -0.096680 -0.259766 0.060303 0.209961 0.503906 0.206055 0.131836 -0.032715 0.163086 -0.302734 0.120117 0.007874 0.028564 -0.149414 0.339844 -0.146484 0.243164 0.008850 0.135742 -0.027222 -0.039062 -0.115723 0.177734 0.046631 0.028442 -0.055908 -0.065918 -0.085449 -0.322266 0.250000 -0.193359 -0.273438 -0.030518 0.216797 0.049805 -0.038086 -0.247070 -0.248047 -0.133789 -0.178711 0.106445 0.213867 -0.142578 0.078613 0.062988 -0.044678 -0.289062 -0.357422 0.275391 -0.062500 0.086914 0.137695 0.074219 -0.042236 -0.173828 0.181641 0.113281 -0.092285 -0.212891 0.031128 0.060547 0.217773 -0.185547 0.028076 -0.054688 -0.090820 0.457031 0.014648 0.228516 0.125000 -0.310547 0.131836 0.239258 0.006958 -0.143555 -0.059814 0.154297 0.084473 -0.090820 0.083984 0.043213 0.138672 0.200195 -0.106934 0.184570 0.190430 -0.042236 -0.185547 -0.134766 -0.333984 -0.010864 0.046631 0.396484 0.089844 0.068848 -0.248047 -0.349609 -0.304688 0.244141 -0.086914 0.014221 -0.090820 0.069824 -0.095703 0.243164 -0.143555 0.017944 -0.157227 0.003662 -0.218750 0.154297 0.200195 -0.078125 0.102051 0.273438 -0.036133 -0.281250 -0.211914 0.108398 0.380859 0.083008 -

0.116699 0.026367 0.100586 0.369141 0.120117 -0.016846 -0.154297 -0.169922 -0.024292 0.005310 0.091309 -0.140625 0.245117 -0.038330 0.233398 0.083008 -0.259766 -0.044678 -0.152344 -0.052734 0.125977 -0.251953 -0.398438 0.316406 0.328125 0.503906 0.380859 0.061279 0.378906 0.024658 -0.152344 -0.328125 -0.080566 -0.218750 0.127930 -0.142578 -0.055664 -0.032959 -0.082031 0.166992 0.068359 -0.181641 -0.088379 -0.099121 0.036377 -0.112305 0.150391 0.126953 -0.184570 -0.231445 -0.194336 0.161133 -0.263672 0.121582 0.049316 -0.245117 -0.044922 0.051514 -0.079102 0.129883 -0.084961 -0.135742 0.324219 -0.009949 0.147461 -0.043945 -0.220703 -0.044189 -0.047607 0.008057 0.170898 0.149414 -0.265625 -0.000645 -0.128906 -0.416016 0.145508 -0.133789 0.173828 0.109375 0.204102 0.083008 0.351562 -0.156250 -0.141602 -0.129883 -0.326172 -0.445312 -0.171875 0.066895 0.116699 -0.198242 0.207031 -0.132812 0.019531 -0.090820 -0.078125 -0.108398 -0.020142 -0.158203 -0.116699 -0.144531 0.136719 0.046387 -0.316406 0.206055 0.062988 -0.132812 -0.287109 -0.257812 -0.118164 -0.109863 0.271484 -0.032471 -0.033691 0.235352 0.030396 -0.083984 -0.235352 -0.023193 -0.119141 0.073730 -0.039307 -0.158203 -0.117188 -0.137695 0.332031 0.163086 0.046875 -0.124023 -0.233398 -0.199219 0.064453 -0.063965 0.100586 0.063965 -0.089844 -0.241211 -0.136719 -0.000248 0.250000 0.013550 -0.142578 -0.187500 -0.109863 0.211914 0.150391 -0.182617 -0.116699 -0.218750 0.339844 0.110352 -0.099609 -0.017456 -0.120605 0.018799 0.094238 0.140625 0.271484 0.115723 0.247070 -0.337891 -0.036621 0.132812 0.020508 -0.166992 0.027832 0.191406 0.073730 0.073730 0.214844 0.017456 0.081055 -0.223633 0.291016]

is: [0.007050 -0.073242 0.171875 0.022583 -0.132812 0.198242 0.112793 -0.107910 0.071777 0.020874 -0.123047 -0.059082 0.101074 0.010742 0.143555 0.259766 -0.036377 0.185547 -0.078613 -0.022705 -0.120605 0.177734 0.049561 0.017212 0.079590 -0.045654 -0.188477 0.189453 -0.023193 0.062988 0.097656 -0.019043 -0.079102 0.152344 0.173828 0.101562 -0.163086 0.114746 0.100586 -0.092773 0.109375 0.058838 -0.021606 0.063477 0.041992 -0.008850 0.032227 0.106445 0.064453 -0.118652 0.030518 0.066895 0.122070 -0.083008 0.171875 0.078613 0.095215 -0.007782 0.023193 0.023438 -0.016846 0.155273 -0.109863 -0.176758 -0.116211 0.023438 -0.010620 0.052734 -0.133789 0.079590 0.073730 0.043945 0.115234 -0.020630 0.074707 -0.011536 0.080566 0.041748 0.080078 0.351562 0.096680 -0.212891 0.165039 -0.078125 0.069824 -0.001396 -0.091309 0.129883 0.251953 -0.016113 0.093262 -0.146484 -0.001511 -0.151367 -0.026855 -0.157227 0.026367 0.085938 0.071777 0.077148 -0.039062 0.054443 -0.127930 0.091309 -0.184570 -0.037598 -0.027954 -0.089844 -0.116699 -0.098633 0.048096 -0.162109 -0.108887 0.084961 -0.045654 0.158203 -0.038086 -0.082031 0.203125 0.086426 0.069336 0.032227 -0.160156 0.094727 -0.024658 0.054199 0.027954 0.044922 0.169922 0.072754 -0.036377 -0.010254 -0.017090 -0.107422 -0.000702 -0.073730 0.253906 0.056641 0.035156 -0.008606 0.185547 0.021484 0.263672 -0.023804 -0.099121 -0.041260 -0.069336 -0.113770 0.050049 -0.058838 0.046143 0.087402 0.105469 0.106445 0.027954 0.094727 0.116211 -0.172852 -0.034912 -0.208008 0.059570 0.104004 -0.001793 0.058594 -0.029785 -0.037598 0.048584 -0.063965 0.079590 0.069336 -0.104980 -0.144531 0.043457 -0.068848 -0.035645 -0.011719 0.013672 -0.065918 0.119141 0.031250 -0.046387 -0.001968 0.007355 -0.056641 0.027832 0.082520 -0.013489 0.071777 0.144531 0.127930 0.042236 0.141602 -0.018066 0.021606 -0.091797 0.133789 -0.195312 -0.050293 -0.037842 -0.096191 0.103027 -0.106934 -0.147461 0.099609 -0.230469 0.227539 -0.075195 0.064941 0.091797 0.046875 0.062988 0.069824 0.046143 0.097168 -0.202148 0.199219 0.186523 -0.119629 -0.142578 0.150391 -0.033691 -0.145508 -0.000690 -0.073242 0.133789 0.035645 -0.022949 0.027710 -0.079102 0.207031 -0.083496 -0.049561 0.031494

0.148438 0.055664 -0.044922 -0.079590 0.004761 -0.020752 0.060059 0.004761 0.011169 0.172852 -0.134766 0.030762 -0.079590 0.090332 0.061035 0.077148 -0.050293 -0.092285 -0.267578 0.107910 0.085938 0.062988 0.107910 -0.026733 0.102051 -0.120605 0.052979 0.094727 -0.165039 0.044189 0.072266 0.041260 0.425781 -0.103027 -0.160156 -0.090332 -0.063965 -0.048096 0.144531 0.065430 0.049316 0.054199 0.135742 -0.019287 -0.215820 -0.074219 -0.146484 0.011475 -0.165039 -0.104980 0.003204 0.134766 -0.003967 -0.103516 -0.139648 0.104492 -0.012573 -0.233398 -0.036377 -0.093750 0.182617 0.027100 0.127930 -0.024780 0.011230 0.164062 0.106934]

here: [-0.027344 0.044922 0.076660 0.133789 -0.083008 -0.056396 0.080566 -0.122070 -0.101074 0.104004 0.051270 -0.085938 0.016113 -0.073242 -0.203125 0.030640 0.273438 0.208984 0.005737 -0.081055 -0.145508 0.083984 0.150391 -0.210938 0.027710 0.043457 -0.013000 0.001740 -0.022827 -0.060059 -0.026978 0.024292 0.010193 0.021118 0.053711 -0.036133 0.002640 -0.055176 0.039307 0.202148 0.056152 -0.056641 0.090820 0.059082 0.017822 0.013306 -0.076660 0.044678 0.043457 0.057861 0.083496 0.122559 0.086426 0.062988 0.024536 -0.158203 -0.010254 0.057861 0.219727 -0.257812 -0.052002 0.009766 0.019531 -0.166016 -0.025757 -0.047607 -0.005737 0.114746 0.051270 0.055908 0.020752 0.027344 -0.056396 -0.078613 -0.094727 -0.031982 -0.022339 0.040771 -0.091309 0.125977 0.017944 0.003281 0.128906 -0.020264 -0.108887 -0.149414 -0.030518 0.132812 0.049072 -0.039307 0.059814 0.143555 -0.130859 -0.085449 -0.316406 -0.010376 0.225586 0.090820 0.019409 -0.104980 -0.173828 0.013916 0.172852 0.116211 -0.084961 0.027222 -0.129883 -0.147461 0.042480 -0.094238 -0.126953 0.068848 -0.064941 0.008179 0.050781 -0.098145 0.221680 -0.051025 0.089844 0.016357 -0.182617 0.022461 -0.129883 0.041748 -0.033447 0.007996 -0.133789 -0.101562 -0.111328 -0.125977 -0.125000 -0.198242 -0.098633 -0.130859 -0.016602 -0.283203 0.059814 0.116211 -0.035645 0.174805 0.128906 0.044434 0.167969 -0.253906 -0.081055 0.073242 0.064941 -0.142578 0.028076 0.001968 0.161133 0.154297 -0.112793 0.066406 -0.010071 -0.040039 -0.002335 -0.030762 0.002029 -0.078125 0.015747 0.027954 -0.050537 0.052002 0.289062 -0.218750 0.055420 -0.037109 0.126953 0.015747 -0.217773 0.123535 0.030029 -0.124512 -0.034912 0.013123 -0.016724 -0.052734 -0.017090 -0.042236 -0.074219 -0.085449 0.083008 0.089355 0.062012 -0.014771 -0.105469 0.085938 0.086426 0.145508 0.024048 0.181641 -0.037109 -0.075195 0.031494 0.001663 -0.020142 0.087402 -0.083496 -0.017090 0.012634 0.149414 -0.068848 -0.007812 -0.030273 0.054688 -0.008606 -0.052979 0.092773 0.156250 0.033936 0.112305 -0.052734 -0.041992 -0.005890 0.147461 0.039062 0.082520 -0.195312 0.096191 0.056396 -0.011108 -0.109375 -0.035400 0.026855 -0.250000 0.076660 0.212891 -0.104980 0.065918 -0.098145 -0.228516 0.103516 -0.010986 0.107422 -0.036133 0.053711 0.067871 0.168945 0.065430 -0.020264 0.043213 -0.045654 -0.015625 -0.125000 0.082520 0.000288 0.103027 0.071289 0.019165 -0.057861 0.081543 0.098145 0.093262 0.093750 -0.051270 -0.010376 -0.023193 -0.229492 -0.167969 0.069336 -0.079102 0.158203 -0.049072 0.118164 0.119629 0.152344 -0.104980 -0.036377 -0.088379 0.116211 0.105957 0.271484 0.075684 0.032227 0.015869 -0.021484 -0.171875 -0.120605 -0.039551 -0.057617 -0.087891 -0.096680 0.078613 0.002213 0.005524 -0.162109 -0.147461 -0.155273 0.080078 -0.096191 0.043457 -0.161133 0.027344 0.049805 -0.015137 -0.011169 -0.125000 0.108887 -0.143555]