Collecting Semantic Information for Locations in the Knowledge Resource of a Text-to-Scene Conversion System

Masoud Rouhizadeh
B.A., The University for Teacher Education, Tehran, 2002
M.A., Allameh Tabatabaie University, Tehran, 2008
Professional Master, University of Trento, Trento, 2009

Presented to the Center for Spoken Language Understanding
within the Oregon Health & Science University
School of Medicine
in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Science & Engineering

November 2013

Center for Spoken Language Understanding
School of Medicine
Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the M.Sc. dissertation of
Masoud Rouhizadeh
has been approved.

Richard Sproat, Thesis Advisor
Research Scientist, Google, Inc.

Jan van Santen, On-campus Advisor
Professor, OHSU

Steven Bedrick
Assistant Professor, OHSU

Aaron Cohen
Associate Professor, OHSU

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0904289.

Contents

List of Tables

List of Figures

Abstract

Collecting Semantic Information for Locations in the Knowledge Resource of a Text-to-Scene Conversion System

Masoud Rouhizadeh

Master of Science
Center for Spoken Language Understanding within
Oregon Health & Science University
School of Medicine

November 2013
Thesis advisor: Richard Sproat, On-campus Advisor: Jan van Santen

WordsEye is a text-to-scene conversion system that receives a text description of a picture from the user via its online interface and converts it into a 3D scene. The core of WordsEye is VigNet, a unified knowledge base and representational system for expressing lexical and real-world knowledge needed to depict scenes from text. In particular, VigNet contains the knowledge needed to map the objects and locations specified in a text into the actual 3D objects. Individual objects typically correspond to single 3D models, but locations (e.g. a living room) are typically a group of objects. Prototypical mappings from locations to objects and their relations are called location vignettes.

This thesis explores our proposed methodology of using Amazon Mechanical Turk (AMT) to populate some portions of VigNet. In the first part, we use AMT to fill out contextual information about VigNet objects, including information about their typical locations and nearby objects, and we filter out Turkers' inputs by WordNet similarity and corpus association measures. Manual evaluation of the Turkers' results show that this is a promising approach.

In the second part, we discuss three strategies for using AMT to collect semantic information for location vignettes. In the first strategy, Turkers describe pictures of different rooms and we then use the WordsEye NLP module to extract the objects in the rooms from their descriptions. In the second strategy, Turkers list the objects that are functionally important for a particular room (such as a sink for a kitchen), and in the third strategy, Turkers name the objects that are visually important, including large objects and furniture. For evaluation, we manually built a set of location vignettes and compare the result of each strategy against that. Our experiments achieved up to 90.62% precision and 87.88% recall.

Chapter 1

Introduction

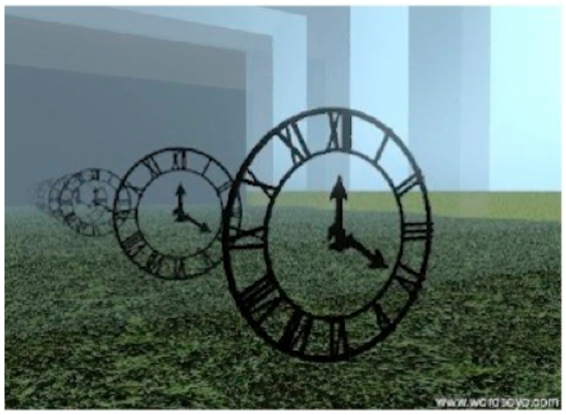
People use language naturally to express their ideas, but sometimes visualizing those ideas into graphics might be a better way to communicate. It is still difficult for many people to produce such graphics, since it is a time-consuming process and requires artistic skills. Moreover, if someone wants to use computer software to generate these graphics, a big challenge will be getting familiar with the software. Text to graphics conversion systems, which receive natural language text input from the user and convert it into the corresponding images or animations, simplify the process of visualizing mental imaginations. These systems do not rely on user's artistic skills or specific graphics software.

Text to graphics conversion systems also have applications in education and computer games. They are great tools for learning new languages. They can help language learners to visualize hence better memorize new words and sentences [11]. They are also helpful in learning grammar and in creative story-telling. In computer games, players can use these systems to interactively construct and modify game environments using natural language input. (See section 2.1 for a review of some well-known text to graphics conversion systems.)

1.1 WordsEye

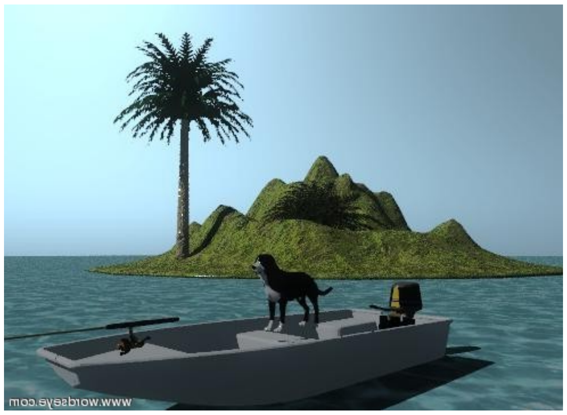
WordsEye[1] [11, 12] is a text to scene conversion system that receives a text description of a picture from the user via its online interface and converts it into a 3D image. To depict a picture from text, the NLP module of WordsEye first parses each input sentence into

[1]www.wordseye.com



Infinite Time by Richard Sproat

The clock is one foot in front of the silver wall. The ground has a grass texture. The texture is one foot wide. A silver wall is two feet in front of the clock. A light is fifty feet above the clock.



The one that got away... by Bob Coyne

The skiff is on the ocean. The grassy mountain is 20 feet behind the boat. The dog is in the boat. The fishing pole is two feet in front of the dog. The bottom of the palm tree is below the bottom of the mountain. It is 20 feet behind the boat.

Figure 1.1: Two pictures generated by WordsEye based on their descriptions.

a dependency structure and processes them to resolve anaphora and other coreferences. The system then uses lexical valence patterns and other information in the VigNet for converting the extracted lexical items and dependency links to semantic nodes and roles. In the next step, the system converts the semantic relations to a set of graphical constraints which represent the position, orientation, size, color, texture, and poses of objects in the scene and finally, it generates the corresponding scene from these constraints and renders that in OpenGL[2] graphic library.[11]. Figure 1.1 shows two pictures created by WordsEye based on their descriptions.

1.2 VigNet

VigNet is the core of the WordsEye system. It is a lexical resource that contains semantic and visual information about the objects, locations and actions that are required for

[2]http://www.opengl.org

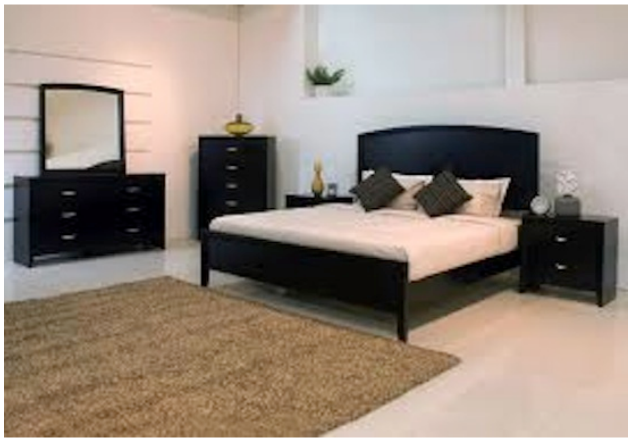
depicting a scene from the input text. For example, if there is a *bedroom* in the input text, to depict that VigNet should contain the knowledge about the typical objects in *bedroom* and their arrangements.

The information in VigNet is represented by a set of objects, and semantic relations between those objects. VigNet contains approximately 18,000 objects such as *flashlight*, *cabinet*, *car*, *wheel*, *spoon*, and *counter* and approximately 4,500 semantic and graphical relations between those objects. Examples of the relations are: *next-to* relation (between *sailboat* and *dock*), *part-of* relation (between *wheel* and *car*), and *on-surface* relation (between *spoon* and *counter*).

1.3 Location vignettes

To convert a text description into the corresponding 3D image, it is necessary to convert objects and locations of the text into the actual 3D objects. This is relatively straightforward for individual objects such as *flashlight*, *sailboat*, and *spoon* since each of these individual objects has a direct mapping to its corresponding 3D object in our library and we can simply use that mapping.

Text descriptions of pictures may also contain words for locations (such as *living room*) which are usually composed of several individual objects. We might expect a *living room* to contain a *sofa*, a *coffee table*, and a *fireplace*. In addition, these objects have a typical spatial arrangement. Perhaps, the *fireplace* is *embedded in* a *wall* and the *coffee table* is *in front of* the *sofa*, in the *middle of* the *room*. Therefore VigNet should contain the knowledge of *the typical objects in each location* and *the arrangements of those objects*. This knowledge is represented in VigNet in the form **location vignettes**. Prototypical mappings from locations to objects and their relations are called location vignettes [36, 35]. Figure 1.2 shows an example of a location vignette for a *bedroom*. This vignette includes a list of the main objects in that pictured bedroom (*bed*, *dresser-1*, *dresser-2*, *nightstand-1*, *nightstand-2*, and *rug*), and the spatial relations between those objects (such as *[nightstand-1] left-side-of [bed]*). Any given location term can have multiple associated



Objects: bed, dresser-1, dresser-2, nightstand-1, nightstand-2

Arrangements:
[bed] against [far wall]
[bed] right-side-of [nightstand-1]
[nightstand-1] against [far wall]
[nightstand-1] left-side-of [bed]
[nightstand-2] against [far wall]
[nightstand-2] right-side-of [bed]
[dresser-1] against [left wall]
[dresser-1] left-side-of [dresser-2]
[dresser-2] against [left-far-corner]
[dresser-2] facing [middle-of room]

Figure 1.2: A bedroom vignette with main objects and their arrangements

vignettes. For example, we can have multiple location vignettes for a *bedroom*, each with a somewhat different set of objects and arrangements.

1.4 Collecting semantic information for locations

In order to build up location vignettes we need to collect information about the typical objects in different locations and their typical arrangements. We also want to collect some contextual information about VigNet objects, such as information about their typical locations and nearby objects. As we will discuss in detail in section 2.2 existing lexical and knowledge resources do not systematically contain such semantic information for locations so we need to build our own lexical resource. One of the well-known approaches to build lexical resources is automatic extraction of lexical relations from large text corpora. As we will see in section 2.3 previous work focused specifically on extracting semantic information for locations, and as Sproat shows in his 2001 work [41] the extracted data from corpora is sometimes noisy and requires hand editing.

In this thesis we propose a new methodology for collecting semantic information for locations by using Amazon Mechanical Turk (AMT). After reviewing the related work in chapter 2, in chapter 3 we discuss about our previous work in using AMT for collecting typical locations and nearby things of objects and then using WordNet similarity and corpus association measures to filter out Turkers' inputs [33, 34]. In chapter 4 we will talk more specifically about the different strategies that we use AMT for collecting semantic information for location vignettes as discussed in two other work [36, 35]. Finally in chapter 5 we discuss the conclusions and the future work.

Chapter 2

Related Work

2.1 Text to graphics conversion systems

There have been numerous other systems for converting text into graphics including [45, 7, 22, 4, 10, 39, 15, 26, 46, 21, 38]. In this report we briefly describe and compare eight well-known systems based on the type of input and output of the systems and their domain and coverage.

To our knowledge the SHRDLU program [45], developed in MIT in 1972, is the earliest system for producing simple graphics from text input. It is also one of the earliest systems to understand and evaluate natural language. The system allows the user to use simple English dialogs to interact with a "robot" living in a closed virtual world. SHRDLU has limited vocabulary and the number of objects and actions is restricted to a pre-existing environment.

Ani [22] is perhaps the earliest text to *animation* conversion system. It creates computer animations from story descriptions. Ani receives descriptions of the personality and appearance of the characters of a story and the interactions among the characters. Then it generates an animation based on the inputs. Ani requires a formal language to describe the characters and their properties and it cannot accept descriptions in natural language. Ani has very incomplete bodies of knowledge and it can only visualize the *"Cinderella"* story.

The Put system [10] is a language-based interactive system to change the spatial arrangements of objects within a virtual image. The input of the system is a formal expression in the form of *"Put Object1, Relation, and Object2"*. An example input can be

"Put the box on the floor". User can choose one of the relations of the system including *"in", "on", "at", "above", "below", "left-of"*, and *"right-of"*. The Put system is limited to pre-existing objects and their arrangements and it only accepts limited number of language expressions as input.

CarSim [15] is a domain-specific system that creates short animations of accident events from written accident reports in French. The CarSim system understands the accident condition by extracting relevant pieces of information from texts such as the type of road, road objects (stop signs, traffic lights, pedestrian crossings and trees), number of vehicles in accident, and sequence of movements of the vehicles. By converting such information into a formal description, the system generates the corresponding 3D accidents scenes showing the movements of the vehicles.

CONFUCIUS [26] is system that receives single natural language sentences and and converts them to multi-modal animation of human characters' actions. SceneMaker [21] is an extension of CONFUCIUS which automatically interprets natural language film or play scripts and generates multimodal, animated graphics from them.

Text-to-Video [38] is a system for generating a visual representation for a short text. The system first parses an input text to generate appropriate and meaningful search terms from that. Using these terms, the system collects some candidate images from online photo collections and then the user selects the final images. After that, the system automatically creates a storyboard or "photomatic" animation.

The main advantage of WordsEye over the other systems is its wide coverage in both linguistic and graphical aspects. It is not limited to a specific domain, it accepts free text input of scene description, and it is the first system that uses a large library of 3D objects to depict variety of scenes. The current system contains 2,200 3D objects and 10,000 images and a lexicon of approximately 15,000 nouns. This speaks to the power and flexibility of VigNet's knowledge representation abilities.

2.2 Location information in lexical resources

As we discussed in chapter 1 we need to include in VigNet, semantic and graphical knowledge about the typical objects in locations and their arrangements, in addition to the information about the typical locations and nearby objects of a given object. Lexical and knowledge resources resources such as WordNet [16], FrameNet [5], Cyc [24], OpenMind [40], and the annotations in the LabelMe [37] project are highly useful resources for semantic, visual, or common sense knowledge but they do not *systematically* contain locational relations that we are looking for. In this section we briefly discuss about these resources and the information they contain about locations.

2.2.1 WordNet

WordNet [1] [16] is an online thesaurus of English developed in 1995 in Princeton. In WordNet, English nouns, verbs, adjectives and adverbs are grouped into sets of synonyms called *synsets*. Each synset expresses a distinct concept, consisting of a set of synonym words (or sometimes a single word), a definition of the concept called *gloss*, and most of the time an example of the usage of the concept in the form of a sentence. Synsets are linked by means of conceptual, semantic and lexical relations and the resulting network of the related concepts can be navigated with the browser. [30].

WordNet does not systematically contain locational information for a given object or location. For instance, under the *bedroom* synset there is no information about what objects are in a *bedroom*, or under the *car* synset, there is no information about the typical locations that we can find a *car* such as *street*, *garage*, and *parking lot*. In a few cases WordNet glosses, contains information about the elements or objects in a location. As an example, the gloss of *bathroom* synset is *"a room ... containing a bathtub or shower and usually a washbasin and toilet"*. It is possible to extract information about the elements of *bathroom* by processing this gloss. However, the number of such entries with these kinds of information is very small and they cannot be used in a principled way.

[1]http://wordnet.princeton.edu/

2.2.2 FrameNet

FrameNet [2] [5] is an electronic lexical resource of English that groups the words together into semantic frames. FrmaneNet contains 10,000 lexical entries and each of the entries is associated with at least one of nearly 800 semantic frames. Each frame represents the joint meaning of the lexical units in that frame. Each lexical unit is also associated with a set of annotated sentences which map the syntactic constituents of the sentences to their frame-based roles. [11]

FrameNet also does not contain detailed locational information. There is a LOCALE frame (with different subframes BIOLOGICAL_AREA, LOCALE_BY_EVENT, LOCALE_BY_OWNERSHIP, LOCALE_BY_USE), that has a frame element constituent_parts. This frame and its subframes include relevant information to what we are looking for but there are only a few sentences that have this frame element in their annotations. In cases that the individual parts of locations are mentioned, the list of parts is not comprehensive and it does contains no visual or spatial relation between the parts. Similar to WordNet, FrameNet in not very informative about indoor locations such as rooms. Most rooms are only annotated using the building_part frame element of the BUILDING_SUBPARTS frame and there is no more semantic or visual information about the elements of objects in the rooms.

2.2.3 Cyc

Cyc[3] [24] is an artificial intelligence project, started in 1984 by Douglas Lenat and it, which tries to put together a comprehensive ontology and knowledge base of everyday common sense knowledge. The goal of the project is enabling artificial intelligence applications to perform human-like reasoning. Cyc knowledge base contains over one million human-defined assertions, rules or common sense ideas and parts of it are released as OpenCyc[4] and ResearchCyc[5] [18].

[2]http://framenet.icsi.berkeley.edu/
[3]http://www.cyc.com/
[4]http://www.opencyc.org/
[5]http://research.cyc.com/

For this work, we have referred to some entries for location in OpenCyc such as the *bedroom* entry. The entry itself contains no information about the objects inside a *bedroom* but the *bedroom furniture* entry contains *bed* as the only furniture in a *bedroom*. As with WordNet and FrameNet, OpenCyc contains only sparse information about the typical objects in rooms. (It should be noted that for this work we did not refer to ResearchCyc which is supposed to include more semantic knowledge. We may examine and utilize this resource for future work.)

2.2.4 OpenMind

OpenMind Common Sense[6] is a system for acquiring common sense knowledge from the general public over the web. The system allows participants to fill in or build natural language templates to express facts, descriptions, and stories. It uses word-sense disambiguation and some other methods for clarifying the collected knowledge and it allows participants to validate knowledge and in turn each other. [40]

The OpenMind Indoor Common Sense[7] is a sub-project of OpenMind and it aims to collect common-sense knowledge about indoor objects for making indoor mobile robots (that work in homes and offices or similar environments) more intelligent [23]. The knowledge base contains information about the objects in an indoor environments, mostly in the form of pre-defined template. As an example, it contains several objects of *bedroom*, represented in templets such as *"You generally find a * in a bedroom."* where *** can be *"pillow", "blanket", "bed", "fan", "dresser", "comforter", "chest of drawers"*, etc.

While such information is in principle useful for our purpose, they are still not precise enough. Since vignettes are abstract representations of locational information, to build them up we are looking for only the *main* objects in each location based on some *visual* or *functional* criteria. This means that we have to filter out many objects of the OpenMind Indoor Common Sense database which we do not assume to be the *main* objects. However, all of these objects can be considered as *potential* objects in each location and this is a useful information for us. We can also use this resource to extract an initial set of the

[6]http://www.openmind.org/
[7]http://openmind.hri-us.com/

typical or potential locations of objects (similar to what we will discuss in chapter 3) but, the extracted information still require manual filtration and annotation.

More importantly, as we discussed in chapter 1 we need the spatial constraints and relations of objects in the locations and the configuration of spatial relations that define a particular location vignette. Such spatial relations are not represented in OpenMind knowledge base and its related resources.

2.2.5 LableMe

LabelMe[8] is a project in MIT to construct a large collection of images with annotated 2D polygonal regions to be used for object detection and recognition tasks. The team have collected a large data set containing many object categories, and multiple instances of a wide variety of images [37].

LabelMe contains well-defined 2D polygonal regions of the annotated pictures and it is a good resource for object detection and recognition tasks. But the 2D polygonal regions are not the exact type of information that we are looking for location vignettes. Moreover, the object list for each location is not precise enough for our task (as with OpenMind). In an annotated set of *bedroom* pictures in the LabelMe dataset we can find objects such as *lamp, bed, painting, curtain, bedstead, bin, capboard, tally, etc.* These objects are obviously good candidate for the object in a bedroom, but not all of those are considered to be the *main* objects in a *bedroom* based on our criteria. As with OpenMind, LabelMe annotations also do not include spatial relations between objects in locations.

2.3 Extracting semantic information from corpora

Considering the fact that we cannot find the exact lexical information that we are looking for in the existing lexical and knowledge resources, we have to develop our own way to collect location information and construct location vignettes. One of the widely used approaches for collecting semantic information and populating lexical resources is extracting semantic information from large text corpora. There has been a large amount of work in

[8]http://labelme.csail.mit.edu/

this filed (For a comprehensive review see [20]). In this section we briefly discuss some of the well-known work.

2.3.1 General approaches

We first discuss the general approaches for extracting different kinds of semantic information from corpora, starting with the work of Vanderwende [44] in 1994, which is a system for automatically interpreting the semantic relations between noun compounds. This system uses relation extraction rules acquired automatically by analyzing the definitions of words in an online dictionary, assigning a weight to each rule, and then applies all of the rules in parallel to determine and rank a set of possible semantic interpretations for each noun compound.

Berland and Charniak [6] present a system for extracting parts of objects from wholes. To find the word P that is possibly a part of the object O, they look for occurrences of patterns like *"P of O"* or *"O's P"* on large text corpora and then sort the possible parts by their statistical association with the objects.

Girju [19] presents an automatic method for detection and extraction of causation relation in English sentences. She reviews a set of patterns for expressing causation relation, such as causative verbal constructions that contain verbs like *cause, lead to, bring about*, and *make*. She presents an inductive learning approach for automatic extraction of these patterns based on their occurrences in annotated text corpora.

In their 2006 paper, Nakov and Hearst [27] use paraphrases or rewritings of noun compounds posed against an enormous text collection as a way to determine which predicates best characterize the the semantic relations that hold within English noun compounds. In their 2007 work [28] they built a system for determining the relations between two nominals, that mines the Web for the sentences that contain the target nominals. The system then extracts the verbs, prepositions, and conjunctions of the sentences and use them as features of a classifier to compare those sentences with the sentences that contain other nominals.

Girju and colleagues [20] describe a system for the automatic identification of a set of seven semantic relations between English nominals based on support vector machines

(SVMs). They used various lexical, syntactic, and semantic features extracted from different sources of knowledge, including lexical semantic resources and annotated corpora, and built a binary SVM classifier for each of the relations based on the features.

Most of this work does not focus on extraction of locational or spatial relations per se and if they do, they mostly focus on extracting *geographical* locations such as the names of countries and cities. Even if they were more concentrated in extracting the locational information of our interest, it was still difficult to distinguish and extract the exact kind of locational information that we are looking for based on our criteria for location vignettes.

2.3.2 Approaches for extracting locational information

Among a few works that specifically focus on such relations, we can name [43, 42], which use the vector-space model and a nearest-neighbor classifier for extracting locational relations. Perhaps the most relevant work to our goal is by Sproat in 2001 [41] in which he uses "likelihood ratios to extract from text corpora strong associations between particular actions and locations or times when those actions occur". This approach sounds promising in some cases. For instance, it correctly assignees *bathroom* as a location to *wash hands*, but the extracted data from corpora is sometimes noisy in some other cases. For example, there were strong associations between the action *eat cheese* and *laundry* since the corpora contain news articles about some criminals who were arrested while eating cheese in a laundry.

In this project we are proposing a new methodology for collecting semantic information for locations by using AMT. In the next chapter we review our approach of using AMT to collect typical locations and nearby things of objects based on our work in [33, 34] and then we will talk the different ways of using AMT for collecting information for location vignettes as we reported in [36, 35].

Chapter 3

Using AMT for collecting the typical locations and nearby objects

To populate VigNet we need to fill out some contextual information about several hundred objects in WordsEye's database, including information about their typical location and typical objects nearby them. This chapter explores our proposed methodology to achieve this goal. First we try to collect some semantic information by AMT. Then, we manually filter and classify the collected data and finally, we compare the manual results with the output of some automatic filtration techniques which use WordNet similarity and corpus association measures on a distributed framework.

3.1 Data collection from Amazon Mechanical Turk

AMT is an online marketplace that provides a way to pay people small amounts of money to perform tasks that are simple for humans but difficult for computers. Examples of these Human Intelligence Tasks (HITs) range from labeling images to moderating blog comments to providing feedback on the relevance of results for a search query [9]. The highly accurate, cheap and efficient results of several NLP tasks have encouraged us to explore using AMT. (For examples of successful AMT tasks in NLP see [9])

We designed two separate tasks to collect information about typical locations of our objects and their typical nearby objects. For **task 1**, Turkers named up to 10 locations in which they might typically find a given object and for **task 2**, Turkers named named up to 10 common objects that they might typically find around or near a given object. We collected 4517 unique responses for locations and 4668 responses for nearby object tasks.

3.2 Pre-processing of the AMT inputs

In the next step we manually checked and corrected the spelling of the inputs for both tasks, and lemmatized them by WordNet lemmatizer module of NLTK[1] (Natural Language Toolkit). In the location task, we compared the inputs against a list of legitimate locations that we prepared before, filtering the inputs that were not present in that list such as *book, painting, poem, dream, movie, and UFO.* We filtered 1522 of Turkers' inputs and finally we came up with 2925 locations for the 257 objects. For the task of nearby objects we did not filter any input and we kept the 4668 nearby objects for 257 of our objects. The data that we collected in this step was in raw format.

3.3 Manual annotation

In the manual annotation process, we (and a colleague[2]) first rejected the undesirable inputs in both tasks. For instance, we rejected *office* as a potential location for a *boat* or *shadow* as a potential nearby object of an *arbor.* Overall, we approved approximately %71.86 of locations and %89.45 of nearby objects. As an example, we approved *clinic, doctor office, emergency room, fire station, firehouse, garage, highway, hospital, military base, nursing home, police station, road, street, trauma center,* and *war zone* for the locations that an *ambulance* can be found and *blanket, car wreck, defibrillator, fire engine, oxygen mask, police, siren, stretcher,* and *vehicle* as examples of potential nearby objects of an *ambulance.*

3.4 Automatic filtering of undesirable data

Manual processing of the data is a time-consuming and expensive approach. As a result, we are investigating different automatic techniques to filter out the undesirable responses from AMT, using current manually annotated data as a gold standard for evaluation of automatic approaches.

[1]http://www.nltk.org/
[2]Margit Bowler

3.4.1 WordNet Similarity measures

In the first approach, we computed some lexical similarity scores for the target and the response items based on the following WordNet (WN) similarity measures. For accessing WN and computing the similarity measures we used the WordNet Interface of NLTK [3]

WN Path Distance Similarity: This score shows how similar two word senses are, based on the shortest path that connects the senses in the is-a (hypernym/hypnoym) taxonomy [3]. We computed this score between each target word and each received response for that target word.

Resnik Similarity: This score shows how similar the two word senses are, based on the Information Content (IC) of the Least Common Subsumer of the two words [3, 31]. Again, We computed this score between each target word and each received response for that target word.

The Average Pairwise Similarity Score: We computed this score based on both WN path distance similarity and Resnik Similarity scores separately. If we assume $W_1, W_2 ... W_n$ to be n responses for target word T; and S_{ij} to be the WN path distance (or Resnik) similarity score between W_i and W_j, then the average pairwise similarity score for W_i will be $\frac{S_{i1}+S_{i2}+...+S_{in}}{n}$. This will provide us the average similarity of each response (i.e W_i) with the other responses (i.e. W_j so that $i \neq j$). In this way we will reward the responses that are more semantically related to each other regardless of their similarity to the target word.

3.4.2 Corpus association measures

The next approach for filtering the raw data was finding the association measure of target-response pairs using Google's 1-Trillion 5-gram web corpus (LDC2006T13) [8], by counting the frequency of each target and response word in unigram and bigram portions of the corpus and then the number of times the two words co-occur within a +/- 4-word window in the 5-gram portion of the corpus. Based on these counts, we used log-likelihood ratio [14] to compute the association between the two words.

MapReduce programming framework: In this work we extended our previous work [33, 34] by counting the word association in Google web corpus using the MapReduce programming model [13] which is an associated implementation for processing and generating large data sets. In MapReduce framework user defines a *mapper* function which processes a set of key/value pairs and *emits* a set of intermediate key/value pairs. Then the *reducer* merges all intermediate values associated with the same intermediate key. MapReduce programs are automatically parallelized and executed on a large cluster of machines. We use MapReduce on Hadoop [2, 1] which is a software framework that supports data-intensive distributed applications and enables applications to work with thousands of cluster nodes and petabytes of data. [17]

Counting word associations in MapReduce: Our word associations algorithm (Figure 3.1) is inspired from word count algorithm. Our Mapper goes through each 5-gram in Google corpus and for each two words within AMT target and inputs (key) that co-occur in a 5-gram string, it emits count 1 (value). Then the reducers accumulates the key-value pairs and sums up all counts for each two-word association. Using this algorithm, computing the word association on approximately 34 million pairs, took 8hrs, 31mins and 52sec.

```
class MAPPER
    method MAP(line a)
        for all term i,j ∈ line a
            If i,j ∈ TargetInputs do
                EMIT(assoc ij , count c)

class REDUCER
    method REDUCE(assoc ij, counts [c1, c2, . . .])
        sum ← 0
        for all count c ∈ counts [c1, c2, . . .] do
            sum ← sum + c
        EMIT(assoc ij, count sum)
```

Figure 3.1: Pseudo-code for the word association algorithm in MapReduce. The mapper emits a key-value pair for each two-word association from target words or AMT inputs (TargetInputs) in a document (5-gram string). The reducer sums up all counts for each association. (Figure adopted and modified from [25].)

Computing associations between target-respond pairs Similar to what we did for WN similarity measures, here we first computed the log-likelihood of associations between the target and the response items. We computed the log-likelihood based based on the counts of all words in the corpus (N), the counts of target word (c_i), the count of the response item (c_j) and the number of times they occur in the same 5-gram in Google web corpus (c_{ij}) using the following equation [32]:

$log(\lambda) = c_i\ log\ c_i + (N - c_i)\ log\ (N - c_i) + c_j\ log\ c_j + (N - c_j)\ log\ (N - c_j) - N\ log\ N - c_{ij}\ log\ c_{ij} - (c_i - c_{ij})\ log\ (c_i - c_{ij}) - (c_j - c_{ij})\ log\ (c_j - c_{ij}) - (N - c_i - c_j + c_{ij})\ log(N - c_i - c_j + c_{ij})$

The average pairwise association score we computed this score based on log-likelihood of two given pairs. As with WN similarity scores, if we assume $W_1, W_2 ... W_n$ to be n responses for target word T; and L_{ij} to be the Log-likelihood score between W_i and W_j, then the average pairwise association score for W_i will be $\frac{L_{i1}+L_{i2}+...+L_{in}}{n}$. This will provide us the average statistical association of each response (i.e W_i) with the other responses (i.e. W_j so that $i \neq j$). Here we reward the responses that are more statistically related to each other.

3.5 Discussion and evaluation of automatic filtration techniques

The collected responses of each AMT task were ranked separately by each of the above similarity and association measures. We classify the ranked responses into "keep" (higher-scoring) and "reject" (lower-scoring) classes by defining a specific threshold for each list. Then we evaluated the accuracy of each filtration approach by computing their precision and recall on correct "keep" items. For each measure and for each list, we examined different thresholds to achieved the best precision without loosing much data (we tried to keep the recall around %50 or higher). Tables 1 and 2 show the accuracy of WN similarity and corpus association measures after applying the close to optimum thresholds. Baseline scores show the accuracy of the responses of each AMT task before using automatic filtration techniques.

	Manual		**WN Path Dist sim**		**Resnik similarity**		**WN Pairwise sim**		**Resnik Pairwise sim**	
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
LOC	71.86	100.0	71.33	48.73	71.61	53.76	72.08	49.38	71.63	49.38
OBJ	89.45	100.0	89.92	56.93	91.49	51.49	90.82	55.01	90.70	49.11

Table 3.1: The percentage accuracy of WordNet Similarity filtering approaches

	Manual		**Log-likelihood**		**Pairwise Log-likelihood**	
	Pre	Rec	Pre	Rec	Pre	Rec
LOC	71.86	100.0	85.27	51.80	74.05	50.38
OBJ	89.45	100.0	92.24	51.00	90.10	55.19

Table 3.2: The percentage accuracy of corpus association filtering approaches

As can be seen in Table 1, none of the WN similarity measures worked well for our task. They hardly could improve the precision, but reduced the recall by around %50. The reason that the WN similarity measures did not help improving precision may be that there is not necessarily a *semantic* relation between different locations of a given object or different object nearby that. The main reason that they reduced the recall maybe that some of AMT inputs are not present in WordNet and as a result we loose them when we use WN similarity measures.

Table 2 shows that we could get good results by using log-likelihood associations between target and inputs in the locations task. We achieved %85.27 precision but we lost nearly half of our inputs (recall is %51.80). Corpus association measures did not work well in the nearby objects task, they slightly improved the precision but they reduced the recall to around %50-55.

Chapter 4

Using AMT to collect information for location vignettes

In this chapter we discuss how we use AMT to collect semantic information about location vignettes. As discussed in chapter 1, each location vignette contains two kinds of information: *typical objects of the location* and *typical arrangements of the objects.* In this report we focus on the task of collecting the main objects of each location and try to examine three different strategies for collecting objects of locations.

In all AMT tasks Turkers required a previous approval rating of 99% and they had to be based in the US. The latter condition increases the probability that the workers are native speakers of English and decreases the chance that they have cultural differences. We presented the Turkers with several pictures of different rooms such as the *bedroom* picture in Figure 1.2. We carefully selected those pictures from the results of image searches from Google[1] and Bing[2] search engines.

4.1 Task 1: Free description of each location

In this task, we had at least two Turkers provide simple and clear descriptions of the pictured room We explicitly asked Turkers that their descriptions had to be in the form of naming the the main elements or objects in the room and their positions in relation to each other. Each description had to be very precise and 5 to 10-sentences long. We also required them to use only “is” and “are” for the verbs of the sentences.

[1]www.google.com
[2]www.bing.com

4.1.1 Extracting location elements from descriptions

In order to extract location information from the free-form descriptions we have obtained from AMT, the text is first processed using the NLP module of WordsEye. (For more details see [35]) We extracted the objects and other elements of locations from processed descriptions which are mainly in the form of RELATION–(attribute)GROUND–(attribute)FIGURE and extract the objects and elements which are represented as FIGURE or GROUND. In other words, the list of FIGUREs and GROUNDs is the list of objects and elements of the given location that the Turkers mentioned in their descriptions. We then further processed this extracted locations as explained in section 4.4.

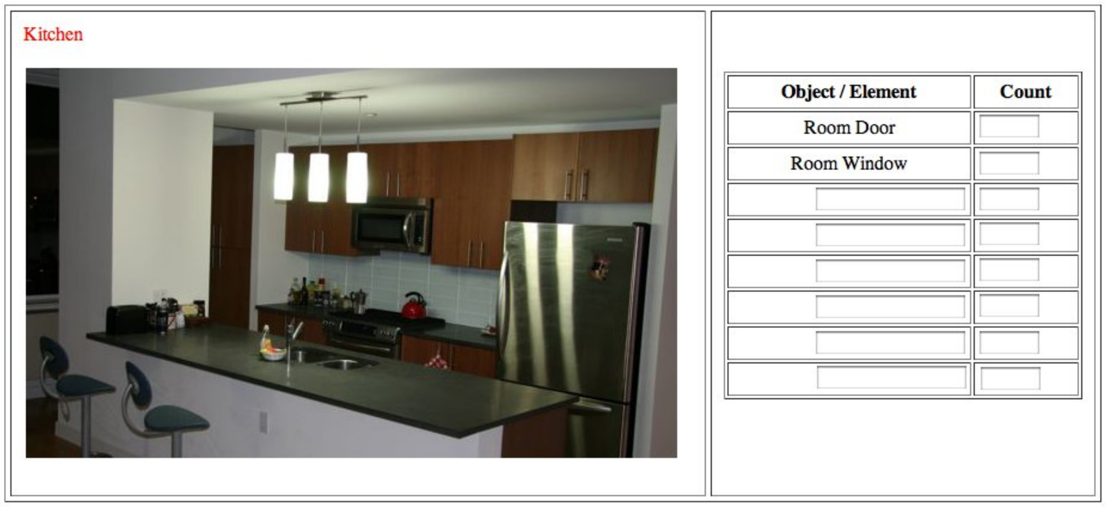
4.2 Task 2: Listing funcionally important objects of locations:

One way to pick up the main objects of a location is to see which objects are *functionally* important for that location compared to the other objects. For example, the important objects for a *kitchen*, are those that are really required in order for the *kitchen* to *be considered* or to *function* as a *kitchen.* Those objects include a *stove*, an *oven*, a *refrigerator*, and a *sink*, but not but a *picture frame.* One can imagine a *kitchen* without a *picture frame* but it is rarely possible to think of a *kitchen* without a *refrigerator.*

Based on this, we designed an AMT task, in which we asked workers to provide a list of functional objects using an AMT hit such as the one shown in figure 4.1. We showed each AMT worker an example room with a list of objects and their counts. We gave the following instructions:

“ Based on the following picture of a **kitchen** list the objects that you really need in a **kitchen** and the counts of the objects.

1. In each picture, first tell us how many room doors and room windows do you see.
2. Don’t list the objects that you don’t really need in a **kitchen** (such as magazine, vase, etc). Just name the objects that are absolutely required for this **kitchen**. ”



Kitchen

Object / Element	Count
Room Door	
Room Window	

Figure 4.1: AMT input form to collect functionally important objects (task 2) or visually important (task 3) objects in locations. Workers are asked to enter the name of each object type and the object count.

4.3 Task 3: Listing visually important objects of locations:

For this task we asked workers to list large objects (furniture, appliances, rugs, etc) and those that are fixed in location (part of walls, ceilings, etc). The goal was to know which objects help define the basic structural makeup of this particular room instance.

We used the AMT input form shown in figure 4.1 again, provided a single example room with example objects and and gave the following instruction:

“ What are the main objects/elements in the following **kitchen**? How many of each?

1. In selecting the objects give priority to:

    - Large objects (furniture, appliances, rugs, etc).
    - Objects that are fixed in location (part of walls, ceilings, etc).

    The goal is to know which objects help define the basic makeup and structure of this particular kitchen.

2. In each picture, first tell us how many room doors and room windows do you see.”

As an example, in the *bedroom* in Figure 1.2 the *bed*, the *dressers*, the *nightstands*, the *mirror*, and the *rug* are visually the most salient objects. They are large and part of the furniture whereas, the *vases* are not large enough to be seen easily and the *pillows* are not part of the furniture. Similar to previous task, here we had Turkers list the objects that are visually important for a location.

4.4 Processing of the extracted objects and the AMT inputs

We collect the extracted objects of Task 1 and the inputs of tasks 2 and 3, and process them in the following steps:

1. Manual checking of spelling: We check the spelling with common spell-checkers in text editors and correct the misspelled words.

2. Lemmatization (plural to singular): here we first change the case of all words to lower-case and then we lemmatize the inputs by the WordNet lemmatizer module of NLTK[3] (Natural Language Toolkit). In particular our purpose of lemmatization is to convert the plural inputs to singular.

3. Removing conjunctions: If the Turkers put multiple inputs in one box (each box is just for one input) separated by "*and*", "*or*", and the slash character ("/") we separate those inputs to multiple objects. For example, we separate the input "*desk and chair*" to "*desk*" and "*chair*".

4. Comparing the inputs agains our 3D object library: We make this comparison to ensure that all the inputs are valid objects based on our object library.

5. Applying a cut-off of three (only for tasks 2 and 3): We find the intersection of the objects and elements from the five inputs provided by five different Turkers. There are a few objects which all five Turkers commonly named and we should have a cut-off of less than five.

[3]http://www.nltk.org/

6. Detecting WordNet synonyms: After collecting the intersections, we further process the non-intersecting items. We see if they are in the same WordNet synset or not, since different Turkers may have used different synonymous words for referring to the same object (such as *tub* and *bathtub*). We add the most frequent word within the synonymous inputs to the list of the intersecting items.

7. Finding major substrings in common: We look for the words that have major common substrings in common. Some input words only differ by a space or hyphen character such as *night stand*, night-stand, and *nightstand.* In such cases, we convert all the variants to the simple with one which has no intermediate character (*nightstand* in this example).

8. Finding direct hypernyms: We see if the two words are direct hypernym of each other in WordNet synset or not. Some Turkers may refer to a more specific or more general concept to refer to the same object (such as *double bed* and *bed*)

9. Looking for head nouns in common: In some cases WordNet does not contain the compound nouns which we are processing. In such cases, we see if the head of the compound noun (the last noun in the compound) can be found in the other inputs or not. As an example, we got *projector screen* and *screen*, both referring to the same object in a classroom picture. We choose *screen* between the two since it is also the head of the *projector screen* compound.

10. Recalculating the intersections (only for tasks 2 and 3): Finally, we recalculate the intersections to collect newly normalized inputs with the cut-off of three.

4.5 Discussion and evaluation of AMT tasks for collecting location vignettes

As one might imagine, evaluation of the results is not an easy task. Sometimes, there is no obvious answer for the question of what are the main objects in a picture. Even if we define clear visual and functional criteria, there might still be disagreement in interpreting those criteria, even among experts. Evaluation of this task is comparable to Machine Translation evaluation in which there is no single answer (as there is in speech recognition) and most of the time, there are multiple correct answers. Considering all these points, we manually built the location vignettes for a set of rooms and used them as our gold standards and compared the results of each tasks against that. Figure 4.2 shows an example description for a *bedroom* and the extracted objects (Task 1), the list of functionally important objects (Task 2), and the list of visually important objects (Task 3).

4.5.1 Using gold standard vignettes to evaluate extracted location elements

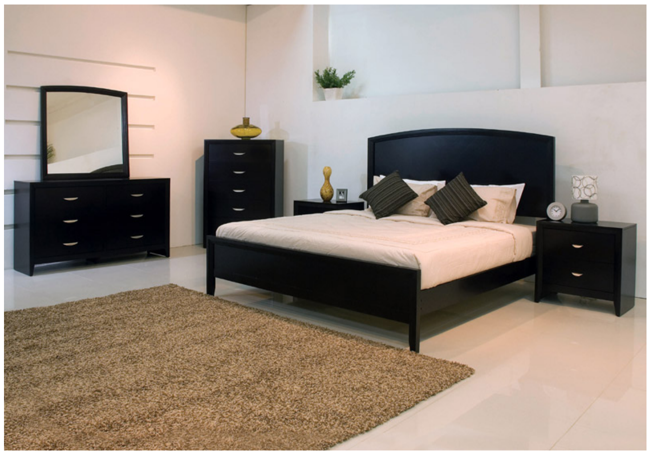
In each gold standard vignette, we have A) a list of objects, and B) the arrangements of those objects. We selected the objects for the gold standard vignettes based on the *visual criteria* i.e. we gave priority to large objects (furniture, appliances, rugs, etc) and those that are fixed in location (part of walls, ceilings, etc). The goal was to select the object that help define the basic makeup and structure of the particular room. Since in this project we focused on the objects of locations and not their arrangements, we compare the extracted objects from the processed inputs with the objects of gold standard location vignettes. (We should mention that the following results are based on manually revised gold standards. We will discuss this in subsection 4.5.5)

4.5.2 Results for the free description AMT task (Task 1)

We extracted 34 objects from the descriptions of 6 rooms. 22 objects were in our gold standard vignettes and this shows that we achieved 64.70% precision. Our gold standard vignettes contain 41 objects and we could extract 22 of those objects, which means we could achieve 53.66% recall.

4.5.3 Results for the functional AMT task (Task 2)

In the task of collecting objects of locations based on functional criteria we extracted 32 objects for 5 rooms from the processed inputs. From those objects, 28 objects were present in our gold standard vignettes which means that we achieved 87.50% precision. Our gold



http://www.gowfb.com/images/Abbyson-Living/Aristo-Bedroom-Set1.jpg

Term: Bedroom

Free description (Task 1):

"The rug is in front of the bed. The bed is between the nightstands. The pillows are on top of the bed. The mirror is above the wide dresser. The plant is above the nightstand. The yellow vase is on top of the tall dresser. The tall dresser is to the right of the wide dresser."

Extracted objects after processing: 'rug', 'bed', 'mirror', 'dresser', 'vase'

Functionally important objects (Task 2):

'mirror' 5, 'dresser' 5, 'bed' 4, 'nightstand' 4, 'pillows' 2, 'mattress' 2, 'rug' 1, 'bed frame' 1, 'lamp' 1, 'comforter' 1, 'chest' 1

Visually important objects (Task 3):

'rug' 4, 'bed' 4, 'nightstand' 4, 'dresser' 4, 'mirror' 3, 'plant' 2, 'lamp' 2, 'pillow' 2, 'area rug' 1, 'clock' 1, 'vase' 1, 'flower pot' 1

Figure 4.2: An example description and the extracted objects (Task 1), the list of functionally important objects (Task 2), and the list of visually important objects (Task 3) provided by Turkers for the pictured *bedroom*. The numbers in front of each objects shows their frequency within the 5 inputs.

standard vignettes contain 33 objects and we could extract 28 of those object from the processed inputs which means we could achieve 84.85% recall.

4.5.4 Results for the visual AMT task (Task 3)

In the task of collecting objects of locations based on functional criteria, we extracted 32 objects for 5 rooms from the processed inputs. From that set of objects, 29 objects were present in our gold standard vignettes that is to say that we achieved 90.62% precision. Our gold standard vignettes contain 33 objects and we could acquire 29 of those object from the processed inputs. In other words we could achieve 87.88% recall.

4.5.5 Manual error analysis and revising the gold-standards

As stated before, the reported results in the above subsections are based on the manually revised gold standards. We made that revisions in the gold-standards based on the following error analysis:

1. Lexical choice: We found out that in a few cases Turkers use different terms from our gold standard terms which were not actually synonyms. As an example, we used the term *end-table* to refer to the small, low table which sits by a bed in the *bedroom* in Figure 4.2, while most of the Turkers preferred to use the term *nightstand* to refer to the same object. In another case we used the term *lectern* whereas most of the Turkers used the term *podium* instead.

2. The visibility of the object: Sometimes Turkers did not mention a partially visible object although it was an important object for the location (such as a *chair* for a *kitchen*). Interestingly, in one case, we did not include an object in our gold-standard for the same reason but the majority of Turkers mentioned that. We then updated our gold-standard to include that object.

3. The importance of the object based on the given criteria: In some cases, the majority of Turkers assumed that an object is important for a particular location (such as a *clock* for a *classroom*) but we did not mention that in the gold-standard. In this particular case, we did not even update the gold-standard since we still think the *clock* was not one of the main object in that particular *classroom*.

4.5.6 Potential objects for each location

As discussed in section 4.4 we applied a cut-off of three on the normalized Turkers' input. In other words, we filtered out the words that were mentioned by less that three Turkers. However, we keep those filtered out words as potential objects of each location. For example, we filtered out *clock*, *lamp*, *pillow*, *plant*, and *vase* in the *bedroom* in Figure 1.2 because of their low frequency, but they are all kept as legitimate potential objects of that room.

Chapter 5

Conclusions and future work

In this work, we investigated the use of AMT for collecting semantic information for locations in VigNet that is the core knowledge base of WordsEye. We introduced the concept of location vignettes that are the prototypical mappings from locations terms to their objects and their arrangements. We had an overview of the well-known text to graphics generation systems and after that, we investigated the possibility of using the existing lexical and graphical knowledge resources to build up VigNet. We also discussed about automatic approaches for extracting semantic information from large text corpora.

Then, we proposed our novel method of collecting semantic information by AMT based on our papers in [33, 34, 36, 35]. In chapter 3 we used AMT to collect contextual information about typical locations of VigNet objects and the typical objects near them, and we applied some automatic filtration approaches using WordNet similarity measures and statistical association, computed on the distributed MapReduce framework. Manual evaluation of the AMT responses (baseline results in Tables 1 and 2) show that we can collect highly accurate data in a cheap and efficient way by using AMT. The accuracy of automatic filtration techniques sounds acceptable as we were able to filter out some undesirable data and improve the precision of the inputs specially in locations tasks. For the future work of this approach, –inspired by [29, 27, 28]– we are planning to use lexical patterns in the Google 1T web corpus in MapReduce framework to extract locational relations between our target words and the AMT inputs using our manual annotations of relations a set of gold standards.

In chapter 4 we investigated the use of AMT for collecting semantic information for location vignettes. In Task 1, we had Turkers describe a picture of a given room and then we extract the elements and objects of the room by syntactic and semantic processing of their descriptions. In task 2 we acquired the objects that are functionally important for a room i.e. the objects that were really needed to define a room, and in task 3, we were looking for the objects that are more visible or they are fixed in the room. We the processed of the extracted objects and the AMT inputs and compared them agains the list of objects in the gold standard vignettes which shows that we can achieve very good accuracy by using the functional and visual strategies (Tasks 2 and 3). Given these very good results, we extended this approach to a set of 85 different rooms.

Location vignettes consist of both a list of objects, and the arrangements of those objects, and we also designed a series of AMT tasks for determining the arrangements of objects in different locations. In that tasks, we use the objects that we collected for each room and we have Turkers determine the arrangements of those objects in that particular room. For each object in the room, Turkers should determine its spatial relation with A) *one wall* of the room and B) *one other object* in the room. For example for the *bedroom* in Figure 4.2, we expect Turkers to enter that the *bed* is *against* the *far wall* and it is on the right side of *nightstand-1.* We force workers to choose the spatial relations among a limited set of relations including *against*, *embedded-in*, *right-side-of* , *left-side-of* , *on*, *in*, *behind*, and *facing.* Similar to the AMT task in chapter 4, we have five different workers to determine the spatial arrangements of objects in each room. The reason that we did not include the results of this task in this report is that we are still exploring the methods for objective evaluation of the *spatial arrangements.* The gold standard location vignettes do include the arraignments of the objects, but we are working on applying this to the Turkers inputs. Similar to section 4.4 we need to post-process the and normalize the inputs of the spatial arrangements to compare them agains gold standard vignettes.

Overall, we have shown here that the vignette semantic approach and our information collection method have great potentials to be extended in populating VigNet. We are planning to extend this to collect vignettes for a variety of indoor locations, including stores, sport complexes, vehicles, etc. and outdoor locations such as streets, roads, beaches, waterfronts, etc. We are also working using AMT in collecting information for *action vignettes* needed to depict different actions from text inputs. Action vignettes are composed of location vignettes, participants of actions, and spatial and temporal relations between the locations, the objects and the participants.

Bibliography

[1] Hadoop–Apache Software Foundation project home page.

[2] *Hadoop in Action*. Manning Publications, 2010.

[3] NLTK WordNet Interface, 2011.

[4] Adorni, G., Di Manzo, M., and Giunchiglia, F. Natural language driven image generation. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics* (Stanford, California, USA, 1984), pp. 495—500.

[5] Baker, C., Fillmore, C., and Lowe, J. The Berkeley Framenet Project. In *Proceedings of the 17th international conference on Computational linguistics* (1998), pp. 86–90.

[6] Berland, M., and Charniak, E. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (College Park, MD, USA, June 1999), Association for Computational Linguistics, pp. 57–64.

[7] Boberg, R. Generating line drawings from abstract scene descriptions. Master's thesis, Department of Electronic Engineering, MIT, Cambridge, MA, 1972.

[8] Brants, T., and Franz, A. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia, 2006.

[9] Callison-Burch, C., and Dredze, M. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk* (Los Angeles, CA, USA, 2010), pp. 1–12.

[10] Clay, S., and Wilhelms, J. Put: Language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications* (1996), 31—39.

[11] Coyne, B., Rambow, O., Hirschberg, J., and Sproat, R. Frame semantics in text-to-scene generation. In *Knowledge-Based and Intelligent Information and Engineering Systems*, R. Setchi, I. Jordanov, R. Howlett, and L. Jain, Eds., vol. 6279 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 375–384.

[12] Coyne, B., and Sproat, R. Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (Los Angeles, CA, USA, 2001), pp. 487– 496.

[13] Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6* (Berkeley, CA, USA, 2004), USENIX Association, pp. 10–10.

[14] Dunning, T. E. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics 19*, 1 (1993), 61–74.

[15] Dupuy, l., Egges, A., Legendre, V., and Nugues, P. Generating a 3d simulation of a car accident from a written description in natural language: The carsim system. In *Proceedings of the ACL 2001 Workshop on Temporal and Spatial Information Processing* (2001), pp. 1—8.

[16] Fellbaum, C. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998.

[17] From Wikipedia, the free encyclopedia. Apache hadoop.

[18] From Wikipedia, the free encyclopedia. Cyc, 2011.

[19] Girju, R. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering* (Sapporo, Japan, July 2003), Association for Computational Linguistics, pp. 76–83.

[20] Girju, R., Beamer, B., Rozovskaya, A., Fister, A., and Bhat, S. A knowledge–rich approach to identifying semantic relations between nominals. *The Information Processing and Management Journal 46*, 5 (2010), 589–610.

[21] Hanser, E., Mc Kevitt, P., Lunney, T., Condell, J., and Ma, M. Scenemaker: Multimodal visualisation of natural language film scripts. In *Knowledge-Based and Intelligent Information and Engineering Systems*, R. Setchi, I. Jordanov,

R. Howlett, and L. Jain, Eds., vol. 6279 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 430—439.

[22] Kahn, K. M. *Creation of Computer Animation from Story Descriptions*. PhD thesis, Computer Science and Artificial Intelligence Lab, MIT, 1979.

[23] Kochenderfer, M. J., and Gupta, R. Common sense data acquisition for indoor mobile robots. In *In Nineteenth National Conference on Artificial Intelligence (AAAI-04* (2003), AAAI Press / The MIT Press, pp. 605–610.

[24] Lenat, D. B. CYC: a large-scale investment in knowledge infrastructure. *Communications of the ACM 38* (November 1995), 33–38.

[25] Lin, J., and Dyer, C. *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool Publishers, 2010.

[26] Ma, M. *Automatic Conversion of Natural Language to 3D Animation*. PhD thesis, University of Ulster, 2006.

[27] Nakov, P., and Hearst, M. Using verbs to characterize noun-noun relations. In *Proceedings of the 12th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA)* (Bulgaria, 2006), pp. 233—244.

[28] Nakov, P., and Hearst, M. UCB: System description for semeval task # 4. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)* (Prague, Czech Republic, June 2007), Association for Computational Linguistics, pp. 366—369.

[29] Nulty, P., and Costello, F. Using lexical patterns in the google web 1t corpus to deduce semantic relations between nouns. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)* (Boulder, Colorado, June 2009), Association for Computational Linguistics, pp. 58–63.

[30] Princeton University. About WordNet, 2010.

[31] Resnik, P. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research* (1999), 95–130.

[32] Roark, B., and Hollingshead, K. Natural Language Processing. [PDF document]. Retrieved from Lecture Notes, 2010.

[33] Rouhizadeh, M., Bowler, M., Sproat, R., and Coyne, B. Data collection and normalization for building the scenario-based lexical knowledge resource of a text-to-scene conversion system. In *Proceedings of SMAP 2010: 5th International Workshop on Semantic Media Adaptation and Personalization* (Limassol, Cyprus, 2010).

[34] Rouhizadeh, M., Bowler, M., Sproat, R., and Coyne, B. Collecting semantic data by Mechanical Turk for the lexical knowledge resource of a text-to-picture generating system. In *Proceedings of International Conference on Computational Semantics (IWCS)* (Oxford, UK, 2011).

[35] Rouhizadeh, M., Coyne, B., and Sproat, R. Collecting semantic information for locations in the scenario-based lexical knowledge resource of a text-to-scene conversion system. In *KES (4)* (2011), A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, Eds., vol. 6884 of *Lecture Notes in Computer Science*, Springer, pp. 378–387.

[36] Rouhizadeh, M., Coyne, B., Sproat, R., and Bauer, D. Collecting spatial information for locations in a text-to-scene conversion system. In *Proceedings of Second Workshop on Computational Models of Spatial Language Interpretation and Generation (CoSLI 2), Boston, MA, July 2011* (Boston, MA, USA, 2011).

[37] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision 77*, 1–3 (May 2008), 157–173.

[38] Schwarz, K., Rojtberg, P., Caspar, J., Gurevych, I., Goesele, M., and Lensch, H. P. Text-to-video: Story illustration from online photo collections. In *Knowledge-Based and Intelligent Information and Engineering Systems*, R. Setchi, I. Jordanov, R. Howlett, and L. Jain, Eds., vol. 6279 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 402—409.

[39] Simmons, R. The clowns microworld. In *Theoretical Issues in Natural Language Processing* (2004), Schank and Nash-Webber, Eds., Association for Computational Linguistics.

[40] Singh, P., Lin, T., Mueller, E., Lim, G., Perkins, T., and Li Zhu, W. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, R. Meersman and Z. Tari, Eds., vol. 2519 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, pp. 1223–1237.

[41] Sproat, R. Inferring the environment in a text-to-scene conversion system. In *Proceedings of The First International Conference on Knowledge Capture* (Victoria, BC, Canada, 2001), pp. 147–154.

[42] Turney, P. D. Expressing implicit semantic relations without supervision. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (Sydney, Australia, 2006), pp. 313–320.

[43] Turney, P. D., and Littman, M. L. orpus-based learning of analogies and semantic relations. *Machine Learning 60*, 1-3 (2005), 251–278.

[44] Vanderwende, L. Algorithm for automatic interpretation of noun sequences. In *Proceedings of the 15th conference on Computational linguistics* (Stroudsburg, PA, USA, 1994), vol. 2 of *COLING*, Association for Computational Linguistics, pp. 782–788.

[45] Winograd, T. *Understanding Natural Language.* Academic Press, New York, 1972.

[46] Ye, P., and Baldwin, T. Towards automatic animated storyboarding. In *Proceedings of the 23rd national conference on Artificial intelligence* (Chicago, Illinois, 2008), vol. 1, pp. 578–583.