國立台灣大學電機資訊學院資訊工程學研究所
碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

多代理人推理系統於整合常識知識之研究

A Multiagent Reasoning System for Commonsense Knowledge Integration

郭彥伶
Yen-Ling Kuo

指導教授： 許永眞 博士
Advisor: Jane Yung-jen Hsu, Ph.D.

中華民國 101 年 1 月
January, 2012

Acknowledgments

「常識，嘗試。」這是這一段旅程的最佳註解，卻不會是句點，只是喘口氣繼續找尋下一個風景。

這段過程中，最感謝的是我的指導教授，許永眞老師。從大四專題開始，老師給予我許多機會去嘗試各種有趣的問題，讓我以初生之犢的姿態接觸到世界上重要的研究，而老師教導我許多面對事情的態度，也會伴隨著我一直走下去。

再者，感謝我的研究夥伴們，Bani、Doudi、Edward，多虧了你們讓我們可以有最活躍的討論，也做出最多有趣的成果，許多事都要感謝你們的幫忙才能順利完成，"Bani 的說話機器人、Doudi 的學習常識的迴圈、Edward 的說故事系統"一定是這過程中最豐富的一站。

感謝婉容學姐和草莓在各方面的協助。也感謝 iAgents 大家的陪伴。

Thanks to Prof. Henry Lieberman for teaching me a lot of things during my visit at MIT media lab. I'll remember the message you give me: "As a computer scientist, we should think about how to make impact to the world."

The most important thing is not only the theories, the systems, and applications we have, but also to use our knowledge to solve the hard problems in our society and the world. From this point, I'll start to tell my own story.

Abstract

Robust intelligent systems require commonsense knowledge. While significant progress has been made in building large commonsense knowledge bases, they are intrinsically incomplete. It is difficult to combine multiple knowledge bases due to their different choices of representations and reasoning techniques, thereby limiting users to one knowledge base and its reasoning methods for any specific task. Instead of merging knowledge bases into a single one, this paper presents a multiagent system for commonsense knowledge integration, and proposes approaches to (1) matching knowledge bases without a common ontology for reasoning, (2) combining different reasoning methods to answer queries from application, and (3) improving coverage of knowledge base via resource-bounded crowdsourcing. Two case studies on video editing and dialog assistance interfaces are also presented to show their improvement in handling user's actions after incorporating the proposed reasoning system.

摘要

智慧系統需要常識知識使其更能應付使用者的各種狀況，雖然現今已建立了許多大型的常識知識庫，但這些常識知識庫也都還不完整。應用程式在使用常識知識時，往往因爲知識庫選擇的知識表現與推理方式而被侷限於只能使用單一的知識庫做應用。相對於將所有知識庫合併成單一知識庫的作法，本論文提出一個用於常識知識整合之多代理人系統，以及系統中的三個重要機制：(1) 配對含有目標知識的知識庫以做推理 (2) 結合多個推理方法來回答應用程式的查詢 (3) 在有限的人力配置下以群眾外包的技術收集知識以增進知識庫的覆蓋率。最後，本論文以影片編輯與對話輔助的使用者介面作爲應用案例，由此兩個案例證實結合此多代理人推理系統的介面代理人可以顯著地增加其處理使用者需求的數量。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an overview of the thesis. First, we explain why an application, especially an intelligent user interface, requires common sense and the problem in using the current commonsense knowledge bases (KBs). We then define the commonsense knowledge integration problem and give our proposed solution to this problem.

1.1 Motivation

Commonsense knowledge is an essential element for building intelligent systems. It enables computers to infer new facts or to perform actions with common sense about the world so that applications can interact with humans intelligently. Also, it helps break the software brittleness bottleneck by taking place whenever the domain-specific knowledge fails [27].

1.1.1 Intelligent User Interface Requires Common Sense

Due to the increasing complexity of the tasks users may perform with an application. The one-to-one correspondence of user interface may not fit user's goal in using an application. Commonsense knowledge has been proposed to bridge the gap between user's goals and functionalities of applications [17]. An interface agent may make use of a broad range of commonsense knowledge, such as event-subevent structure, temporal relationship, and specific facts, to provide assistance to users [18].

Consider we are browsing media items on the web to gather information about "the earthquake that struck Japan on March 11, 2011." There are great difficulties in having a comprehensive understanding of this event using the current media browsing interfaces. Most of the interfaces provide only single view to list media items. Figure 1.1 (a) and (b) sort news according to timeline and location respectively. We need to browse through every item in the collection to find out the desired items. The search or recommendation functionalities are only keyword spotting and recommendation by producers (see figure 1.1 (c) and (d).) For example, when we searched for videos about "Japan tsunami" using the current search engine, we got no video mentioning the victims in the earthquake. These problems limit user's experience in media browsing. The interface agent with commonsense knowledge would act as an assistant to organize the media items using the context of an event, rather than as a conventional tool to list content.

These contextual commonsense knowledge is useful not only for organizing media items but also for context-aware and storytelling interfaces.

Figure 1.1: Traditional media browsing: (a) Map view. (b) Chronological view. (c) Key word search. (d) Video suggestion by producer.

1.1.2 Challenges in Using Common Sense

For thirty years, many projects [16, 39] have been devoted to the collection of commonsense knowledge. While significant progress has been made in building large commonsense KBs, e.g. Cyc [16] and ConceptNet [11], such KBs are still intrinsically incomplete or even inconsistent. An interface agent easily fails to provide assistance to users due to these problems. Therefore, it may be necessary for an intelligent system to reason with multiple commonsense KBs at the same time in order to meet the specific goals of an applications. Example 1 demonstrates a sample scenario from an application developer's viewpoint.

Example 1. (Goal-oriented search engine [19])
The goal-oriented search engine is a search engine interface that uses commonsense reasoning to turn search goals specified by the user in some natural language description into effective query terms. When processing an input like "my golden retriever has a cough", it should identify that the user's search goal is to find a veterinarian/remedy for his/her dog. Unfortunately, systems developed using ConceptNet alone will be unable to find the answers, as it does not contain any knowledge about golden retrievers. Alternatively, an intelligent system can first consult the lexicon database WordNet to find a generalization of the concept, e.g. "golden retriever is a kind of dog". It is then possible to find "veterinarian" by reasoning from "dog" and "cough" in the ConceptNet semantic network.

In summary, enabling applications to reason across multiple KBs improves their goal-achieving behaviors. In the dynamic world today, it is especially important

that interface agents should utilize the up-to-date knowledge in multiple KBs to interact with their users.

1.2 Problem Definition

Using knowledge from multiple KBs at the same time can be viewed as a process to ask an knowledge integration system to reason out the answer to a given query. This section offers the formal definitions of commonsense knowledge integration, including the commonsense KB and format of input queries used in this paper.

1.2.1 Notations

Before giving the notations, we first present the scenario in which developers use commonsense knowledge to build applications. A *commonsense knowledge base* that contains sentences acquired from *acquisition agents* (e.g. experts, online users, or text mining algorithms). Developers use *input queries* to ask knowledge base for the knowledge required in applications. Different *reasoning methods* can be applied on one knowledge base to reason out answers to different queries. Developers use *input queries* to ask KBs for the knowledge required in applications.

Commonsense Knowledge Base

Suppose there are n available commonsense knowledge bases. Let $\mathcal{K} = \{K_1, K_2, \ldots, K_n\}$ be the set of available KBs and each KB contains sentences in its own representation and is a subset of world facts. We use $I_n = \{i_{n1}, i_{n2}, \ldots, i_{nm}\}$ to denote the set of

reasoning methods that can be performed on knowledge base K_n.

Input Query

An input query q is used for finding contextual information of a concept. The first part of input query is the targeted concept, which is represented in natural language. The second part specifies the in/output type of a query. The in/output type is denoted by type selected from three categories: noun phrase, verb phrase, and adjective phrase (see table 1.1). For example, we use "bible:object;location" as our input query if we would like to find the related locations of the object "bible."

Table 1.1: Categories of concept type

Category	Types
Noun phrase	actor, object, time, and location
Verb phrase	action and goal
Adjective phrase	adjective

1.2.2 Commonsense Knowledge Integration Problem

Since the queries that can be answered by a KB forms the basis of an intelligent system, the commonsense knowledge integration problem can be considered as the following formulation:

Definition 1. Given a set of input queries $Q = \{q_1, q_2, \ldots, q_l\}$ and commonsense KBs $\mathcal{K} = \{K_1, K_2, \ldots, K_n\}$ with different reasoning abilities $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$, a knowledge integration system should answer the queries Q by combining or interoperating different commonsense KBs. The knowledge integration problem is

constructing a knowledge integration system that can answer more queries than the system that uses any single KB alone.

Same as the answers returned by a single KB, the answers returned by the knowledge integration system can also be used by applications. Using the API provided by the knowledge integration system, application developers are able to get answers to their queries without learning usages of multiple commonsense KBs. As a result, the application that adopts the knowledge integration system as its base would be more robust in handling users' requests.

A number of tools have been proposed to tackle the knowledge integration problem by merging existing ontologies or KBs into a single one, including PROMPT [29], FCA-Merge [45], CHIMAERA [22], and Blending [10]. However, previous study [13] on extending Chinese ConceptNet by "Blending" with English ConceptNet showed that it often brings many noises into the merged knowledge base. It is hard to apply these merging method to large and noisy commonsense KBs crowdsourced from general public.

1.3 Proposed Solution

Instead of merging KBs into a whole by mapping multiple ontologies, my idea is to *combine different reasoning methods* to handle queries that cannot be answered by a single KB. By using other reasoning methods as an intermediate step to continue the inference chain, we are able to answer more queries.

In order to handle the heterogeneity of commonsense KBs, we choose a multia-

gent framework to build the commonsense knowledge integration system. Multiagent systems provide a powerful paradigm to facilitate application building when multiple heterogeneous knowledge representations and reasoning methods are required [26]. The proposed system will satisfy the following requirements:

- **Allow heterogeneous knowledge representations of KBs**

  Each research group builds their own commonsense knowledge base according to their purposes. Hence, it is hard to find a unified schema to impose on these KBs. We should utilize the reasoning abilities of KBs while maintaining their own autonomy.

- **Facilitate cross-KB reasoning**

  Answering more queries using commonsense knowledge bases involves the interoperation of different reasoning methods in multiple KBs, as described in example 1. By organizing reasoning methods in different ways, more queries can be answered to enhance the abilities of applications to handle user's requests.

- **Reflect changes after updates of KBs**

  The uptodate commonsense knowledge bases are collected from the online users or automatically extracted from web pages. This kind of knowledge bases evolve and change faster than KBs codified by knowledge engineers. The system should also identify and reflect changes in its reasoning results.

The agents in my proposed system would *match suitable KBs*, *compose their reasoning methods*, and *monitor KBs* to handle queries. Chapter 3 will give a

detailed discussion about the proposed system and design of each agent.

1.4 Thesis Organization

This thesis start by overview of commonsense computing technologies a nd the related projects for knowledge/service integration. Along with the definision of the multiagent framework used in our commonsense integration system, the mechanisms for matching KBs, composing reasoning methods, and improving coverage of KBs are introduced. We then present three experiments to evaluate the performance of each mechanism. Two case studies are conducted to discuss how the proposed system can be integrated into the existing interface agents to provide more assistance to users. This thesis concludes with a summarization of the proposed system and future work.

Chapter 2

Background

2.1 Commonsense Computing

For many years, several groups have been devoted to building commonsense knowledge bases. Despite large number of facts have been collected for different knowledge bases (see table 2.1), they are intrincically incomplete and inconsistent. For example, the overlap of concepts in ConceptNet and WordNet is only 4.79%.

Table 2.1: Number of concepts in different knowledge bases

Knowledge base	Number of concepts
ConceptNet	274,477
WordNet	128,391
Wikipedia	3,440,143

In order to fully utilize common sense in programs, a system to integrate different commonsense knowledge bases are required for the benefit of application building. However, it is hard to integrate different knolwedge bases into a whole due to their

different design decisions in representations, quantity, quality, and means of access. This section reviews the essential elements in commonsense computing to demonstrate the heterogeneity of commonsense KBs.

2.1.1 Knowledge Representation

When building applications, developers may choose commonsense knowledge bases with different knowledge representations to serve their specific requirements [39]. The two most prominent representations for common sense are formal logical framework and semantic network, used by Cyc [16] and ConceptNet [20] respectively.

The formal logical framework is appropriate for representing precise and unambiguous facts, which facilitates the automation of commonsense reasoning. On the other hand, the semantic network is more flexible in incorporating new knowledge and contextual reasoning. It represents all sentences in the corpus as a directed graph [11]. The nodes of this graph are *concepts*, and its labeled edges are *relations* between two concepts. For example,

- `UsedFor(a, b)`, e.g. [Spoon] is used for [eating].
- `IsA(a, b)`, e.g. [Dog] is an [animal].

2.1.2 Commonsense Knowledge Collection

Codifying millions of pieces of human knowledge into machine usable forms has proved to be time-consuming and expensive. While techniques for mining knowledge from corpus or web pages have been developed [8, 36, 2], it is difficult for computers

to discover the commonsense knowledge underlying a text [7]. Therefore, sources of commonsense knowledge are still majorly reliant on experts or the general public.

Expert-developed Knowledge Bases

A team of knowledge engineers encode common sense into the knowledge base. This approach ensures the highest quality of data. However, it is expensive, time-consuming, and difficult to scale up.

WordNet WordNet [25] is a highly structured database of words, which are carefully crafted by expert linguists. Synonyms are grouped into *synsets* and are conneted with each other by relations. It has been successfully used in a variety of applications to measure the proximity of words.

Cyc Started in 1984, the Cyc project [16] carefully crafted knowledge into CycL, a rigorous logic-based language to ensure its correctness. Now, the OpenCyc 2.0 ontology contains hundreds of thousands of terms with millions of assertions relating the terms to each other.

Collaboratively-built Knowledge Bases

With the success of Web 2.0 sites and crowd-sourcing techniques. Many research groups start to use websites or games to appeal online users for contribution. This approach helps the collection of large amount of data. However, the data collected from these sources is highly dependent on the performance of users, which also makes

the knowledge bases incomplete and inconsistent.

ConceptNet The Open Mind Common Sense (OMCS) project at MIT [40] has collected over a million sentences in multiple languages and encoded them into semantic network. The English and Portuguese corpora were collected from over 15,000 contributors at the OMCS website[1] within the span of 10 years. In addition, about 20% of the English sentences were collected via Verbosity, a human computation game [46]. With innovations in community-based social games, the uptodate knowledge in the Chinese ConceptNet was successfully collected and verified via question-answering between players within a year [15].

Wikipedia Wikipedia[2] is one of the world's largest knowledge bases of both encyclopaedic knowledge and commonsense knowledge. The knowledge is stored in documents connected with page links. It also provides a taxonomy by its categories, where articles can be assigned to one or more categories. The unstructured documents are thus put into a network of categories.

2.1.3 Commonsense Reasoning

It is straighforward to equip a variety of applications with common sense by querying the knowledge bases using APIs. For example, one may ask if a specific assertion is present in the corpus. Furthermore, knowledge bases with different representations may call for different reasoning methods.

[1] http://openmind.media.mit.edu/
[2] http://www.wikipedia.org

Commonsense Reasoning in Semantic Network

The semantic network is suitable for finding related and similar concepts. Measures of similarity/relatedness quantify how much two concepts are alike/related. Both relatedness and similarity measures are developed in WordNet [32], ConceptNet [43], and Wikipedia [44] so that it is possible to reason in large and noisy semantic

AnalogySpace [43] generalizes the reasoning method called *cumulative analogy* [3] so that it is robust enough in large and noisy semantic network. The assertions are divided into *concepts* and *features*, i.e. descriptions of concepts such as "UsedFor eating" or "dog IsA". The knowledge in ConceptNet is represented as a sparse matrix, and its most prominent features can be identified by using singular value decomposition (SVD). Concept similarity is defined in terms of their shared features. Figure 2.1 is the projection of the first two dimensions of Chinese ConceptNet. We can find that the proximate concepts shares some features. For example, the concepts on the 1st axis is the things people don't want.

Blending [10] was proposed as a technique to integrate common sense into other systems. In particular, blending of commonsense knowledge with domain-specific knowledge can be done by finding an analogical closure across multiple, previously separated sources of data. Two sparse matrices are combined linearly into a single, larger matrix. Reasoning with blended knowledge bases containing overlaping information can produce inferences that would not be produced from either input alone. The blended knowledge bases are often used for identifying the prominent senses of a domain-specific knowledge. For instance, applications have been developed to show how the blended knowledge bases help affect sensing [1] and word

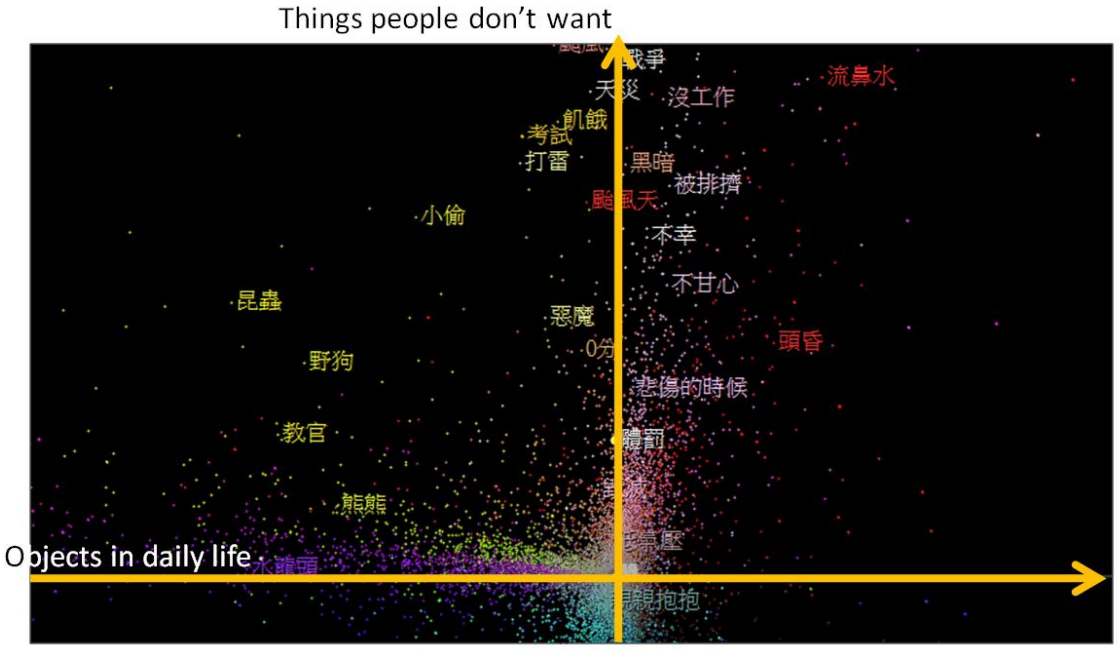


Figure 2.1: The projection of Chinese AnalogySpace

sense disambiguation [12].

These reasoning techniques can also be integrated into our reasoning composition as one of our reasoning method to provide reasoning for a specific domain. With more reasoning methods involved in our integration system we are able to produce various reasoning results for applications.

Commonsense Reasoning in Formal Logical Framework

The logic framework, on the other hand, uses deduction and theorem prover to reason new facts. Heuristics are often applied to logic-based reasoning for better efficiency. OpenCyc[3] also released its planner for reasoning out actions and events

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

with its rules and assertions. These reasoners are exploited in Cyc's intelligent assistant [31] to recognize user's plan. However, these logical reasoning aims to provide the exact answer to users. Given the different reasoning properties, the logical reasoning methods are rarely apply on the contextual reasoning required by interface agents.

2.2 Related Work

Multi-agent systems have been proposed to integrate and reuse information in web environment for many years. Most of them either integrate information from different sources for clients to query or combine the existing web services to create new funtionalities for applications to use. This section reviews approaches to integrate mutliple knowledge sources and services.

2.2.1 Knowledge Source Integration

Most of the knowledge integration systems opt to combine loosely coupled information sources into integrated wholes, such as KRAFT [33] and KSNet [42]. In such systems, it is often necessary to merge existing ontologies into a single ontology. Ontology mapping [9] is a process that combines distributed and heterogeneous ontologies based on linguistic or structural similarity. Multiple knowledge sources can then be reused and queried based on the merged ontology. InfoSleuth [28] used another approach to integrating heterogeneous information sources. An application-specific ontology is used as a basic ontology to locate different queries. However,

these approaches are not feasible for commonsense knowledge integration, because there is no common ontology known a priori. There also exists conflicts in different commonsense knowledge bases, e.g. the confidence of being true of an action is different in different contexts, such that it may result in errors if we merge them into a single one.

2.2.2 Web Service Composition

Instead of combining different information sources, *service composition* [38] puts multiple services together to create new funtionalities. This approach is more flexible for applications to use the desired services in dynamic and heterogeneous web environment.

Some initiatives, e.g. Web Services Business Process Execution Language (WS-BPEL) [30] and OWL-S process model [21], are defined to represent service composition where the data/excution flow of a composite service is known. However, the depndency of data and process may not be available because web services are developed by different organizations and can be created on the fly [35]. Dynamic composition method aims to generate the composition plan automatically. Similar to the context of planning problem, each web service can be specified by its preconditions and effects such that a composition plan can be generated automatically by planners without knowledge of predefined workflow. Several techniques such as situation calculus [23], rule-based planning [24], Hierarchical Task Network (HTN) planner [41], and logical theorem prover [34] have been used in web services composition.

In our commonsense knowledge integration problem, the reasoning methods provided by different knowledge bases can be viewed as different web services. In order to facilitate interoperation of commonsense knowledge bases, we can also combine multiple reasoning methods to perform complex reasoning while the targeted parts are not available in one knowledge base. The state-of-the-art service composition techniques are plausible ways to combine existing reasoning methods.

Chapter 3

Commonsense Knowledge Integration

This chapter describes the multiagent framework and mechanisms used for answering queries by composing different reasoning methods. Before giving detailed description of the mechanisms, I would l ike to introduce the framework used in this thesis.

3.1 Multiagent Framework

Figure 3.1 presents the proposed multiagent framework for commonsense knowledge integration. Instead of merging knowledge sources into a single ontology, the key idea of this framework is to treat knowledge as resources that reasoning methods can access to provide services for applications. The integration of knowledge bases is achieved via matchmaking and composition of different reasoning methods.

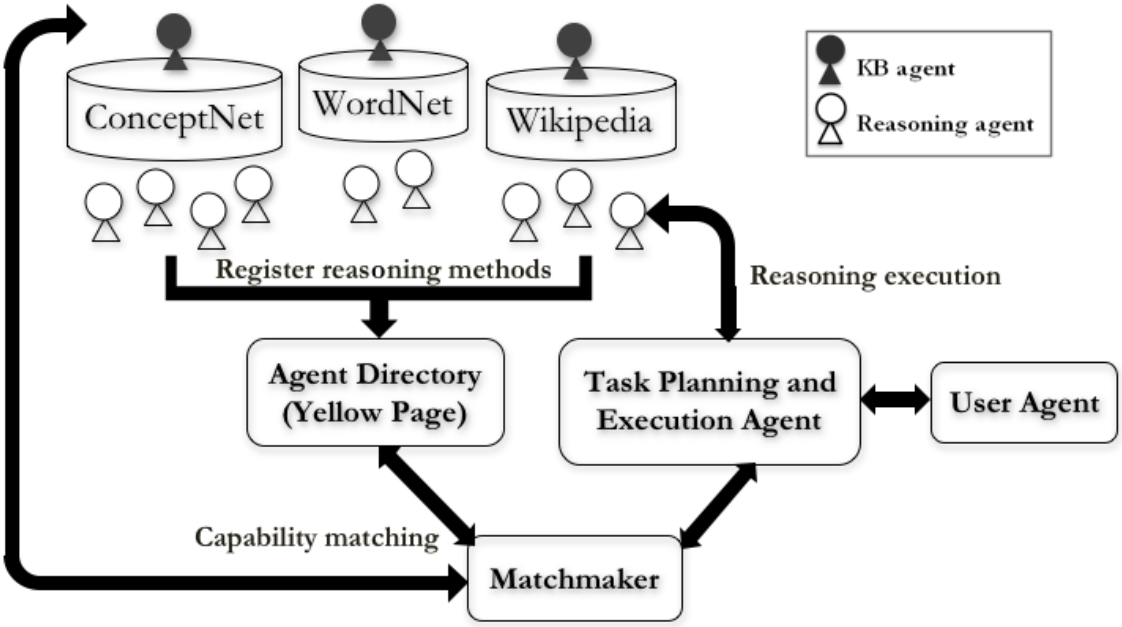


Figure 3.1: The multiagent framework for commonsense knowledge integration

3.1.1 Agents

Each knowledge base is maintained by a KB agent and is used by reasoning agents. Dependencies between reasoning methods are managed and aggregated by task planning agents. Following are the details about the responsibilities of the agents in this framework.

- **KB agent**: A KB agent monitors knowledge contained in a KB by creating a capability model of the KB. If a matchmaker asks for its capability of handling a query, it will answer according to the capabilty model of the KB. It is also equipped with behaviors to improve the coverage of a KB.

- **Reasoning agent**: There are multiple reasoning agents for a KB. They are building blocks for answering a query. Each reasoning agent performs an atomic task that corresponds to a reasoning method of the KB such as finding related objects of an activity. Once the KB agent updates the KB, the quality of reasoning result is also improved.

- **Agent directory**: An agent directory records the types of reasoning agents in the system and the knowledge bases they can access. It provides the information of reasoning agents for the matchmaker to sort out the potential matches.

- **Matchmaker**: A matchmaker forms sample queries to KB agents to check which knowledge bases are able to handle the incoming request. It returns a list of matched reasoning agents which are sorted by their capability of answering

the query.

- **User agent**: A user agent represents a specific application. In most applications, it is an interface agent. It observes users' actions and forms queries to the task planning agent to get reasoning results to provide assistance to users.

- **Task planning and execution agent**: Based on the query of a user agent, a task planning and execution agent sends the requests to the matched reasoning agents and forms a composite inference chain to answer query.

3.1.2 Mechanisms

There are three required mechanisms in realizing such a framework. First, it needs to match the reasoning agents that access KBs containing the targeted knowledge domain. Unlike service matchmaking in service-oriented computing, there is no common ontology for matchmaker to identify the capability of large commonsense KBs with heterogeneous knowledge representation. Second, a composition sequence of reasoning agents should be dynamically generated to find out the desired output. The profiles of reasoning agents should be specified to facilitate composition. Finally, the system should maintain a good reasoning quality. In recent knowledge acquisition processes, online users are often recruited to contribute their common sense. However, previous research also found that unguided crowd-sourcing suffers from high redundancy and contributes little to the coverage of KB [4, 14]. Since the coverage of a KB is a key factor in improving the query answering ability of a knowledge-based system, a KB agent will designed to improve the coverage of a KB

by utilizing other KB as a guide. In the following sections, I'll describe the models and procedures used in the three mechanisms.

3.2 Matchmaking of Reasoning Agents

The capability of a reasoning agent can be described by the scope of knowledge it can access. In order to decide whether a reasoning agent can handle a request from task planning agent properly (i.e. return good answers), each KB agent should be equipped with a compact model that summarizes its KB, and be able to inform the matchmaker of its evaluation results quickly. This section introduces our distributed capability model which is built by transforming the concepts contained in the KBs into a k-dimensional space using low-rank approximation. Queries from applications are evaluated using vector similarity to decide which KB to match up with.

3.2.1 Capability Modeling

To most KBs, we can argue that they can be determined by or associated with a small set of eigenconcepts. Therefore, we can use the eigenconcepts as the capability model of KB agent. For example, latent semantic analysis (LSA) [5] shows the empirical success of this argument.

Correlation Matrix

Consider a grounded commonsense knowledge base K, a sentence can be represented as a triple, $(c_i, \mathit{relation}, c_j)$, where c_i and c_j are concepts. Each concept can form a

vector of related concepts $\vec{v}$, where the j_{th} component of the vector, v_j, is the number of triples containing c_i and c_j in K. Thus, we can construct an $n \times n$ *correlation matrix* A, where n is the number of concepts in K. The correlation matrix reflects the capabilities of a KB agent by describing the relatedness of concepts in the KB it can access.

Knowledge Represented by Eigenconcepts

In order to quickly evaluate whether a KB can answer a given query, we need to reduce the dimension of a correlation matrix without losing its capability. The high-dimensional concept correlation matrix is then re-formulated into a k-dimensional vector space, where $k \ll n$. A concept in K is represented as a vector in a k-dimensional space spanned by *eigenconcepts*. It is the responsibility of a KB agent to identify the dimensions that are useful in summarizing the KB while truncating dimensions that are less relevant.

Low-rank approximation is an approach to achieving dimensionality reduction. Given a $n \times n$ correlation matrix A, we aim to approximate it by a matrix of rank k, which is much smaller than n. In this thesis, we apply singular value decomposition (SVD) on the correlation matrix to achieve low-rank approximation: $A \approx A_k = U_k \Sigma_k U_k^T$, where

- U_k: a $n \times k$ matrix that relates concepts to eigenconcepts
- Σ_k: a $k \times k$ diagonal matrix of singular values σ_i that assigns weights to each eigenconcept.

The best rank k is chosen in algorithm 1 so that the resulted vector space is the best approximation to describe the correlation of concepts in the KB. We call this vector space "the capability model of the KB."

Algorithm 1 Capability Model (K)

Require: A commonsense knowledge base K
Ensure: A set of eigenvectors that span the capability model U_k and the projection A_k of concepts in K
1: Build a correlation matrix A from triples in K
2: Apply SVD on the correlation matrix: $A = U\Sigma U^T$
3: Choose the largest k such that $\sigma_k - \sigma_{k+1} \leq \theta$ where θ is a small constant
4: Represent concepts in the k-dimensional capability model: $A_k = U_k^T A$
5: **return** U_k and A_k

The confidence of choosing k by $\sigma_k - \sigma_{k+1} \leq \theta$ is motivated by the Eckart-Young theorem [6].

Theorem 1. (Eckart-Young Theorem) [6]
Let $A = U\Sigma V^T = U\,diag(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)V^T$. For any k with $0 \leq k \leq r$,

$$\|A - A_k\|_2 = \sigma_{k+1} \tag{3.1}$$

Theorem 1 implies that $\sigma_k - \sigma_{k+1}$ is the key factor of incorporating the k_{th} column of U into the compact representation of capability model. If we set θ small enough, we are returned with the representative eigenconcepts so that the error between real and modeled capability of KB is within σ_{k+1}.

3.2.2 Capability Evaluation for Matchmaking

With the capability model, a KB agent is now able to answer whether it contains sufficient knowledge for its reasoning agents to answer the requests. Here, two kinds

of concept vectors are used for capability evaluation:

- **Knowledge-based concept vector**, a_c: The knowledge-based concept vector, a_c, is the projection of concept c in A onto the k-dimensional capability model, where $a_c = U_k^T A^{(c)}$ and $A^{(c)}$ is the column of concept c in A. We use this vector to represent the knowledge domains involving concept c in the KB.

- **Application-based concept vector**, v: For an application that would like to incorporate commonsense knowledge, it should identify a corpus or a set of concepts to describe the knowledge domain of the application. If the application requires up-to-date news, we may use Google search snippets to describe its knowledge domains. Then, a concept vector v can be created from the co-occurred concepts of concept c in the corpus. We use this vector to state the knowledge domains required by the application.

The KB agent evaluates its capability of answering a query regarding concept c by comparing whether the its knowledge-based and application-based concept vectors are aligned in the capability model. In the evaluation procedure, the KB agent projects input application-based vector v onto its capability model, i.e. $v_k = U_k^T v$. The capability of KB to handle the request is correspondent to the similarity between a_c and v_k. The higher the similarity score of a_c and v_k, the higher capability the KB has to provide the required knowledge for its reasoning agents to answer the query. Algorithm 2 illustrates the procedure a KB agent performs to evaluate the capability of the monitored KB. The evaluation score returned by algorithm 2 is defined by the cosine similarity of a_c and v_k.

Algorithm 2 Evaluate Capability (U_k, A_k, c, v)

Require: A capability model U_k, a projection A_k of concepts in K, a query concept c, and a application-based concept vector v

Ensure: An evaluation score *sim* that indicates the capability of K to handle a request

1: Project v onto the capability model: $v_k = U_k^T v$
2: Get vector of c from A_k: a_c = column of c in A_k
3: Calculate similarity of a_c and v_k: $sim = \frac{a_c \cdot v_k}{\|a_c\| \|v_k\|}$
4: **return** *sim*

Once the matchmaker gets the request from the task planning agent, algorithm 3 is used for matching suitable KBs. The matchmaker asks every KB agent to get the evaluation score of each KB respectively, and then sort the KBs based on their evaluation scores in descending order. Finally, the KBs with evaluation scores > 0.5 are returned to the task planning agent. The task planning agent will send requests to the reasoning agents of the matched KBs to get answers.

Algorithm 3 Matchmaking $(\mathcal{K}, c)$

Require: A set of available KBs $\mathcal{K}$ and a query concept c

Ensure: A sorted list of matched KBs L

1: Construct application-based vector v
2: **for** K_i in $\mathcal{K}$ **do**
3: sim_i = Evaluate Capability (U_{ik}, A_{ik}, c, v)
4: **end for**
5: **for** K_i in $\mathcal{K}$ **do**
6: **if** $sim_i > 0.5$ **then**
7: Add KB K_i into L
8: **end if**
9: **end for**
10: Sort L in descending order based on the evaluation scores of KBs
11: **return** L

3.3 Reasoning Composition

The planning and execution agent forms inference chain using multiple reasoning agents if it fails to retrieve answer from any of the reasoning agents. However, it is hard to generate a composition plan dynamically without any knowledge of the reasoning agents.

Consider the problem of combining multiple reasoning agents to get the target reasoning results. It is hard to automatically generate a composition plan over the commonsense semantic networks without any knowledge of the reasoning methods. With a meta description of reasoning agents, the composition can then be converted to a planning problem. This section presents our proposed profiles for reasoning agents and a planning procedure that uses these profiles to dynamically plan a chain to target concepts. We choose logic planner to manipulate the inference chains, and the knowledge in commonsense KBs to interact with users.

3.3.1 Profile of Reasoning Method

Before giving the algorithm to find the contextual information of a target concept, we describe profile used in the reasoning composition process. The profile of reasoning method serves as a hint for composition.

Profile

A reasoning method can be viewed as a service provided by a commonsense KB. Therefore, ontologies for sevice profiling, e.g. OWL-S [21], can also be used for spec-

ifying the reasoning method. To differentiate the reasoning methods, we consider the attributes listed in follows are the properties a reasoning agent should specify so that the planner can use these information to compose reasoning chain.

(1) Input concept: An input concept comes from user agents or the output of other reasoning agents. Every reasoning agent takes an input concept to perform an atomic reasoning task.

(2) Input type: Since concepts may have multiple senses, the input type serves as a way for sense disambiguation so that the reasoning method can provide accurate inference result.

(3) Output type: An output type attaches to every concepts output by reasoning methods. With the type of output concept, we can check if it is our desired output. If it is not our target output type, we may redirect the output concepts to other reasoning methods to form complex reasoning.

(4) Access KB: In most cases, an atomic reasoning method accesses only one commonsense KB. Different commonsense KBs may pervide the same reasoning function for applications. For instance, both WordNet and ConceptNet can generalize a concept using their own hypernym/IsA relation.

(5) Language: The up-to-date commonsense KBs always support multiple languages. In order to activate a reasoning method, the language it supports should match the request from interface agent.

Concept Type

The input/output types used in the profile of reasoning agents follow the ontology defined in figure 3.2. In order to keep reasoning agents connected to natural language, the representation user agents used to interact with users, the concepts are associated with three major types: *noun phrase*, *verb phrase*, and *adjective phrase*. The user agent can use a shallow parser and its application domain to determine the type of an input concept. The type of an output concept is determined by the reasoning agent that generate the output concept. This kind of abstraction simplifies the usage of natural-language text and turns natural-language input into a more computationally useful form for logic planner.

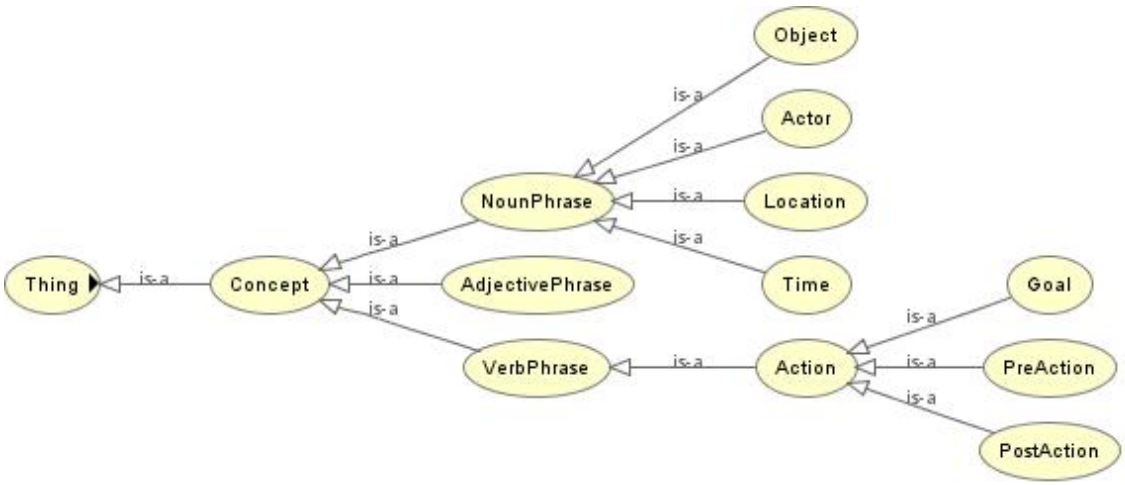


Figure 3.2: The ontology of concept types

3.3.2 Composition Algorithm

Given an input query and the available reasoning agents, the reasoning composition process can be considered as a planning problem. The input and output types are the pre-conditions and post-conditions of a reasoning method. This thesis uses a rule reasoner to achieve composition.

The composition rules and input/output concepts of reasoning agents form a KB of reasoning methods. In the rule-based planning, a reasoning agent is activated only if the input concept and its profile satisfy the rules. Two rules are defined for the basic composition of reasoning methods. They are listed in follows:

(1) If there is a new input concept that is not used by any reasoning agent and the input type and language of a reasoning agent matches the new input concept, we can infer a new instance of the reasoning method and use the new input concept as its input.

(2) If the type and language of an output concept matches the input type and language of a reasoning method, we can infer a new instance of the reasoning method and use the output concept as its input.

Procedure of composition is illustrated in algorithm 4. In each iteration, the task planning agent asserts the output concepts into the KB of reasoning profiles and infers instances of reasoning methods using the composition rule defined in this section. This process does a breadth-first traversal with multiple commonsense semantic networks until the type of output concepts matches the target output type or the expected relevancy of the inference result is low.

Algorithm 4 Composition (q)

Require: A knowledge base K_r containing the profiles of reasoning agents $\mathcal{R}$, a input query concept q with type t_i, and an output concept type t_o.
Ensure: A list of related concepts L.

```
1: L ⟵ ∅; d = 0;
2: Assert input concept q with type t_i into K_r;
3: R = Infer possible reasoning methods in K_r;
4: repeat
5:    C = Get reasoning results from inferred reasoning methods R;
6:    for each concept c in C do
7:       if type of c matches t_o then
8:          Add c into L;
9:       else
10:         Assert c into K_r;
11:      end if
12:   end for
13:   d = d + 1;
14:   if Evaluate Relevency (L, C, d) then
15:      R = Infer possible reasoning methods in K_r;
16:   else
17:      break;
18:   end if
19: until L ≠ ∅
20: Sort concepts in L by relevancy;
21: return L
```

Consider an example in which our input concept "golden retriever" is an "actor" and the expected output type is "location". Figure 3.3 shows part of the composition results. First, golden retriever is asserted in the KB of reasoning profiles to activate "location finder", "action finder", and "generalizer". Even if the "location finder" fails to find the related location of golden retriever, other output concepts "chase frisbee" and "dog" are asserted by other reasoning methods and activate "location finder" to find output locations. The evaluation function in algorithm 4 is used to

estimate the relevancy of the retrieved concept to decide whether should we continue the inference chain or not.

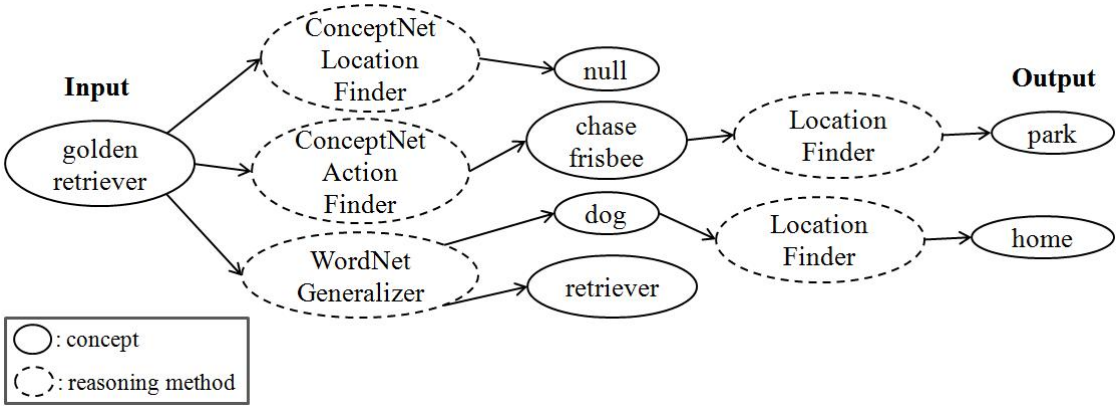


Figure 3.3: Example of reasoning composition

From the example, we can find that this composition process inherently facilitates the interoperation of commonsense KBs since we use reasoning methods as the building block for composition.

3.4 Improving Coverage of KB by Crowd-sourcing

Crowd-sourcing of commonsense knowledge can be viewed as a process to ask online users for sentences that can be put into the knowledge base. In general, the coverage of facts in a KB is limited by the human and/or computational resources available for knowledge acquisition. A KB agent should utilize the available resources to improve the coverage of KB. In this section, I review the knowledge acquisition problem through a question answering framework and then introduce the *resource bound* in knowledge acquisition. With the consideration of available resources, it

is possible for KB agent to improve coverage of KB by acquiring more informative sentences.

3.4.1 Resource-bounded Knowledge Acquisition

Before giving the algorithm to acquire knowledge, we first present the question answering framework used in this section. In addition to the definition of KB and reasoning methods mentioned in chapter , *questions* are used in crowd-sourcing for asking acquisition agents for *answers* to add into the KB.

Knowledge Base

As defined in chapter , K is a knowledge base that contains grounded sentences and is a subset of world facts. Given a set of concepts C and a set of relations R, every sentence is a triple (s, r, o), where $s \in C$, $o \in C$, and $r \in R$. Every triple in K must be associated with either *true* or *false*. If an inference algorithm i can derive sentence μ from K, we write $K \vdash_i \mu$.

Question and Answer

As in Learner [3], we use the complement of a concept in a sentence as a feature f of that concept. We use F to denote the feature set. A concept s in triple (s, r, o) is associated with a right feature f_{right}. Similarly, an concept o in triple (s, r, o) is associated with a left feature f_{left}. Therefore, a question q is asking acquisition agents for the associated concepts of a feature such as "(_, `IsA`, basketball player)?" We use $\mathcal{Q}$ to denote the set of possible questions whose size is $2|C||R|$. For any question

set $Q \subseteq \mathcal{Q}$, let $Answer(Q)$ be the answers returned by a set of acquisition agents $\mathcal{A}$ and $Answer(Q) \subset \mathcal{X}$. If no agent has answer to the questions, $Answer(Q)$ would be an empty set $\emptyset$.

Knowledge Acquisition

Since the sentences in a knowledge base forms the basis for the reasoning process of an intelligent system. The acquisition of commonsense knowledge can be considered as the following formulation:

Definition 2. Given a commonsense knowledge base K, the knowledge acquisition problem is using $Q \subseteq \mathcal{Q}$ to find a set of sentences $Answer(Q)$ such that $|\{\alpha : (K \wedge Answer(Q)) \vdash_i \alpha\}| > |\{\alpha : K \vdash_i \alpha\}|$. If no such question set Q exists, the K contains sufficient knowledge for programs to use.

Definition 2 implies that we aim to add new sentences to K instead of the sentences that are already in K or can be inferred from K. The most intuitive method to find such sentences is to try all the possible questions, i.e. $Q = \mathcal{Q}$. Therefore, it always takes a lot of time to ask acquisition agents all the questions if we build a knowledge base from scratch.

Resource Bound

In building large commonsense knowledge base, it is impratical to enumerate all the possible questions ($|\mathcal{Q}| = 2|C||R|$, this number is over millions for existing facts) since the KB agent can only use a fixed amount of resources to get answers. *The*

resource limitation in knowledge acquisition comes from the number of acquisition agents we can access and the computing power these agents have. For example, the knowledge base constructors only have a fixed amount of money to recruit a fixed number of workers on Amazon Mechanical Turk[1] and each of them can answer at most n questions within a short period of time. To quantify the resource limitation in the knowledge acquisition process, we compute the number of questions that can be answered by the acquisition agents in time T:

Definition 3. Given a set of available acquisition agents $A = \{a_1, a_2, ..., a_m\}$ and the number of questions $N = \{n_1, n_2, ..., n_m\}$ they can answered in time T, the *resource bound* Θ in the knowledge acquisition process is $\sum_{i=1}^{m} n_i$.

Note that n_i in definition 3 is a small constant compared to the number of possible questions, i.e. $n_i \ll |\mathcal{Q}|$. Since there is always only a fixed number of agents involved in the crowd-sourcing of common sense, the resource bound Θ is also far less than $|\mathcal{Q}|$.

With the resource bound Θ, we cannot put all the possible questions into the question set Q to get a good enough $Answer(Q)$. So, the knowledge acquisition problem turns to the resource-bounded case as the following definition:

Definition 4. Given a commonsense knowledge base K and a resource bound Θ, the resource-bounded knowledge acquisition problem is using $Q \subseteq \mathcal{Q}$ to find a set of sentences $Answer(Q)$ such that $|\{\alpha : (K \wedge Answer(Q)) \vdash_i \alpha\}| > |\{\alpha : K \vdash_i \alpha\}|$ within time T where $|Q| \leq \Theta$

[1] https://www.mturk.com

In the resource-bounded case, we aim to find a question set Q whose size is smaller or equal to Θ such that the acquired answers infer more new sentences for the knowledge base. With no assumption or background knowledge of the knowledge domains, the question set could be any combination of $2|C||R|$ possible questions. If we randomly choose one combination of questions, it is easy to get $\emptyset$ or the sentences already in K. Many people try to find a productive question set Q to elicit new sentences from agents. However, most of them are heuristics and require a lot of domain knowledge or skills to codify the question set. It is hard to apply these heuristics to another domain or large commonsense knowledge bases developed from general public.

3.4.2 Guiding KB

Instead of using heuristics to reduce the size of question set, we think the key component of finding a productive question set is to estimate the answers of a question and their inference results before asking for answers from acquisition agents. Using the estimated answers and inference results, we are able to put the most productive questions into the question set so that we can get most new sentences within the resource bound to improve the coverage of knowledge base.

Consider the problem of estimating the answers and their inference results. If we have another knowledge base containing the target domain knowledge and it shares some concepts and relations with K, we can use the sentences and inference results in another knowledge base as plausible answers to the questions. Therefore, we can compute the differences of the two knowledge bases to find a productive question set

Q such that we can elicit answers from agents to fill in the gaps. We say that this process is *knowledge base approximation* and another knowledge base is a *guiding knowledge base* K_g.

3.4.3 Similarity as a Weak Inference

Before giving the algorithms to find a productive question set, we illustrate the inference algorithm used in the knowledge base approximation process.

KB Matrix

In order to compare the answers and inference results with other knowledge bases, we put all the sentences in K to a "KB matrix" where the $(i, j)_{th}$ entry of the matrix is the truth assignment of the sentence that is consisted of i_{th} concept in C and j_{th} feature in F. For example, table 3.1 is a sub-matrix of a KB matrix.

	`IsA`(_, pet)	`AtLocation`(_, home)	`CapableOf`(_, fly)	`MadeOf`(_, metal)
cat	True	True	False	False
dog	True	?	False	?
airplane	False	False	True	True
toaster	?	True	?	True

Table 3.1: The example of KB matrix.

Semantics of KB matrix For any two rows i, j in the KB matrix, we can find that the sentence (c_i, f) in an inference rule can be replaced by sentence (c_j, f) and gives plausible inference results if c_i and c_j have the same truth assignments for

the same feature. For example, the sentences $PartOf(fur, cat)$ and $IsA(cat, pet)$ in modus ponens rule

$$\frac{PartOf(fur, cat) \rightarrow IsA(cat, pet), PartOf(fur, cat)}{IsA(cat, pet)}$$

can be replaced by $PartOf(fur, dog)$ and $IsA(dog, pet)$. After the replacement,

$$\frac{PartOf(fur, dog) \rightarrow IsA(dog, pet), PartOf(fur, dog)}{IsA(dog, pet)}$$

is still a plausible inference since dog and cat share the same truth assignments for features $PartOf(fur, ?)$ and $IsA(?, pet)$.

From the above observation, we can find that two concepts will get involved in the same inference rules if they have shared features and vice versa. In this thesis, *the similarity of two concepts is defined with respect to the shared features of the two concepts*. Therefore, we can transfer the inference results of the concept which is similar to our targeted concepts instead of doing the complete inference for every sentence in the knowledge base. Using similarity as a weak form of inference, we are able to perform the same inference procedure in different knowledge bases. The KB matrix is a good fit for computing similarity of two concepts.

Similarity Measure for Large and Noisy KB

For large commonsense knowledge bases that are constructed by crowd-sourcing techniques, the confidence of a sentence in knowledge base may not be 100%. Therefore, we relax the constraint of truth assignments of KB matrix to real numbers, which indicate the confidence of being true for the sentences and are in range $[-1, 1]$.

Since the size of knowledge base is always very large, the KB matrix must be large and sparse. As in AnalogySpace [43], we apply truncated singular value decomposition (truncated SVD) on KB matrix to smooth the noisy data in the knowledge base. The concepts are then transformed to a k-dimensional vector space spanned by eigen-features. In the vector space spanned by eigen-features, the proximity of two concepts represents their level of overlaps in features. Therefore, the similarity of two concept vectors $\vec{c_1}$ and $\vec{c_2}$ is defined as follows:

$$Sim(\vec{c_1}, \vec{c_2}) = \frac{\vec{c_1} \cdot \vec{c_2}}{\|\vec{c_1}\|\|\vec{c_2}\|}$$

3.4.4 Acquisition via KB Approximation

Now that we have similarity measure for any two concepts in a knowledge base, we can find the inference results of a concept. By measuring the differences of inference results in K and K_g, we are able to create questions that would draw new sentences for K to fill up the gaps.

Concept Coverage

In order to compare the inference results of two knowledge bases, sub-KB matrices are created with the glossary mapping of concepts/relations in the two knowledge bases. The inference results of a concept are reflected on its similar concepts because the similar concepts are plausible to involve in the same inference rules. If the top n similar concepts of a concept is the same in different knowledge bases, we say that the *coverage* of the concept is the same in the two knowledge bases. If its coverage

is different in two knowledge base, we can always find a new sentence to improve the concept coverage. The coverage of each concept is evaluated using algorithm 5 where the number n of similar concepts is specified because we only compare the similar concepts.

Algorithm 5 Concept Coverage (K, K_g, Map, n)

Require: A knowledge base K, a guiding knowledge base K_g, a mapping $Map : K \rightarrow K_g$ of concepts/realtions of the two knowledge bases, and n to indicate the number of similar concepts

Ensure: Coverage score of target concepts

1: Use the domain and range of Map to create KB matrices M and M_g of K and K_g
2: **for all** concept $c \in M$ **do**
3: Get top n similar concepts of c in M to form set S
4: **for all** concept $c_i \in Map(c)$ **do**
5: Get top n similar concepts of c_i in M_g to form set S_i
6: **end for**
7: Coverage score of $c = \frac{|(\bigcup_i S_i) \cap S|}{|S|}$
8: **end for**

Question Set Generation

For the concepts with low coverage scores, we borrow the features from its mapped concepts in guiding knowledge base K_g to generate new questions. The answers to the generated questions are guaranteed to cover at least the sentences found in K_g. Using algorithm 6, we include the questions generated from the concepts with lowest coverage scores into Q and ensure that the size of Q is within the resource bound Θ. Therefore, the generated questions are the ones that would draw answers to approximate the inference results in the guiding knowledge base.

Algorithm 6 Generate Questions (K, K_g, Map, Θ)

Require: A knowledge base K, a guiding knowledge base K_g, a mapping $Map : K \rightarrow K_g$ of concepts/realtions of the two knowledge bases, and a resource bound Θ

Ensure: A question set Q and $|Q| \leq \Theta$

```
1: Sort the concepts according to their coverage scores and store them into a list L
2: Create KB matrix M and M_g of K and K_g
3: for all concept c ∈ L do
4:    for all concept c_i ∈ Map(c) do
5:       Get features F of c_i in M_g but not in M
6:       for all feature f ∈ F do
7:          if |Q| ≥ Θ then
8:             return Q
9:          end if
10:         Get relation r from f
11:         Create question q using c and r
12:         Add q to Q
13:      end for
14:   end for
15: end for
16: return Q
```

Chapter 4

Experimental Design and Result

In order to evaluate the proposed mechanisms for the commonsense knowledge integration system, we deviced three experiments to see how these mechanism performs. In the following sections, we'll first introduce our implementation of the system, and then evaluate the system from three aspects: correctness of matchmaking, improvement in quality of reasoning results, and coverage improvement of crowd-sourced KB.

4.1 Experimental Setup

The commonsense knowledge integration framework is implemented using JADE (Java Agent DEvelopment Framework). ConceptNet, WordNet, and Wikipedia are chosen to be our experimental knowledge bases. We used the ConceptNet and WordNet APIs to implement the atomic reasoning methods.

4.1.1 Reasoning Method

The reasoning methods form the basis for reasoning composition. In our prototype system, six reasoning methods are used for providing contextual information of an input query. They are generalizer, similar concept finder, activity finder, actor finder, location finder, and object finder.

Two generalizers are implemented to find the upper class or general description of a concept. They use "hypernym" and "IsA" relation to generalize a concept in WordNet and ConceptNet respectively. A similar concept finder outputs concepts of the same type as input concept. As in AnalogySpace [43], a similar concept finder represents concepts in English ConceptNet as a high dimensional vector space. The similar concepts are then retrieved from the proximate concepts of its input concept. For other reasoning methods, like activity/actor/location/object finder, they utilize different relation types in ConceptNet to retrieve the related subevents/actors/locations/objects of an input concept. All of these reasoning methods reflect the structure of commonsense semantic network in their inference procedure.

4.1.2 Planner

The profiles of aforementioned reasoning methods are stored in a predefined ontology that follows the OWL-S specification. With the rule reasoner provided by Jena[1], a semantic web framework for java, the composition rules are written in Jena rules and used for inferring new instances of reasoning methods. In contrast to the reasoning

[1] http://jena.sourceforge.net/

in large commonsense KB, it is much more efficient to do reasoning in small ontology of reasoning profiles.

4.2 Matchmaking of Reasoning Agents

In order to evaluate the matchmaking results, we incorporated the proposed capability model into the system. The matchmaking results produced by our approach are compared with the matches made by online users.

4.2.1 Experimental Design

In this experiment, snippets from google's search results are selected as our corpus to create the application-based concept vector v in the matchmaking. Related concepts of a specific query term are returned by reasoning agents of the matched KB.

Collecting Matchmaking Lists From Online Users

In order to evaluate the matchmaking results, we collected a user-generated matching list as the ground truth. The list was collected from workers on Amazon Mechanical Turk, the largest crowd-sourcing market in the world. First, we uniformly sampled 100 concepts from each knowledge base as input queries from user agents. About half of these concepts were found in at least two knowledge bases. Every worker in our task was given a concept and three web pages containing related concepts of that concept. The web pages are generated from the three knowledge bases. What the worker required to do was to browse the web pages and rate the relatedness of the

web page and the given concept. Every concept was rated by 3 workers to increase its validity. After this process, we created a ranking of knowledge bases for each query concept according to their relatedness with the query concept. The knowledge base ranked first was considered as the matched knowledge base for finding related concepts.

Building Correlation Matrix

Since the correlations of concepts are in different forms for different knowledge bases, we had to use different method to create the triples required by KB agents. Triples in ConceptNet are its assertions; triples in WordNet are its words and the relations between words; triples in Wikipedia are the pages and their links with other pages. The θ in algorithm 1 is set to 0.1 for every KB agent.

4.2.2 Experimental Result

In this experiment, our matchmaking mechanism used the evaluation scores of the query concept in each knowledge base to create a ranking of knowledge bases for each query concept. Accuracy and rank correlation are two measures we used for evaluating matchmaking correctness and relevance against the user-produced ranking.

Correctness

Intuitively, the correctness of a matchmaking mechanism is whether it can find user's desired result. If the result ranked first in the matchmaking is also the knowledge

base ranked first by online users, we marked it as a "match". The accuracy is thus defined as the proportion of matches to query terms:

$$accuracy = \frac{\#\ of\ matches}{\#\ of\ query\ concepts}$$

For all sampled concepts, the accuracy is 93.32%. If we only consider the concepts that can be found in at least two knowledge bases, the accuracy is 87.67%, which is slightly lower than the former case. This drop of accuracy appears in the concepts with polysemy. For such concepts, it is likely that the application-based vector may not be aligned with the user's goal, therefore causing errors in matching knowledge bases. It indicates that with application-based vectors corretly representing an applications' goal, the matchmaker can produce an accurate match using the proposed capability model.

Table 4.1: Experimental results: Accuracy and Kendall's τ rank correlation coefficient

	Accuracy	τ_B
All concepts	93.321%	0.818
Concepts existing in at least 2 KBs	87.671%	0.695

Relevance

We also compare the rank correlation of the matchmaking produced by the matchmaker and online users. The relevance of our matchmaking results and ranked list produce by online users were measured by Kendall's τ rank correlation coefficient τ_B. Given two lists of same length, let n_c be the number of concordant pairs (i.e.

pairs that are ranked in the same order in both rankings), n_d be the number of discordant pairs (i.e. pairs that are ranked in opposite order in the two rankings), x_0 be the number of ties that exists only in the matchmaking ranked list, and y_0 be the number of the ties that exists only in the ideal ranking list. Then, the Kendall τ rank correlation coefficient for the existence of ties is defined as:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_c + n_d + x_0)(n_c + n_d + y_0)}} \tag{4.1}$$

If the two ranked lists are in perfect agreement, τ_B is 1; if they are perfect disagreement, τ_B is -1. In our experiment, the average τ_B are 0.818 and 0.695 for all query concepts and query concepts in at least two knowledge bases respectively. A positive correlation was found between the two lists. This result suggests that the matchmaking result produced by our capability model is correspondent to users' ranking of KBs.

4.2.3 Discussion

In this experiment, we also discovered some good properties of the capability model:

- **Reflect the complexity of knowledge representation**

  The capability model of different knowledge bases can be further analyzed by the dimensions of their capability model. The k selected by algorithm 1 is 31 for ConceptNet, whereas the k for WordNet is 11. This difference results from the complexity of knowledge representations. The structure of WordNet is much simpler than that of ConceptNet since most of the relations in WordNet

are (*concept1*, `IsA`, *concept2*) triples. Using the capability model, it is easy to classify concepts into a particular categories in WordNet.

- **Can be used for modeling expert knowledge**

  Wikipedia also contains expert knowledge in its pages, e.g. history, theory, or formula of an item. When we use the links in pages to build the capability model, we include both common sense and expert knowledge into the capability description of Wikipedia. This extension suggests that the capability model can also be used for modeling knowledge bases of very specific domains since it requires only the correlation between concepts to build the model. The evaluation score will be adjusted according to the correlation we consider in the model. For example, the evaluation score for "Stuttgart" (a city) is higher than that of "bridge" in Wikipedia, because geography and history descriptions are common in expert knowledge of a city but are not so common in the descriptions of commonsense knowledge like "bridge".

These properties allow us to apply this model to other KBs so that we can utilize not only commonsense knowledge but also expert knowledge in our integration system.

4.3 Reasoning Composition

In the experiment of reasoning composition, we random sampled 750 concepts from English ConceptNet. Every sampled concept is attached with four output types: "actor, location, action, and object" to form 3,000 input queries. These queries are answered by three reasoning approaches: 1) directly query English ConceptNet,

2) directly query the merged KB of ConceptNet and WordNet, and 3) reasoning composition proposed in this thesis. Both the coverage and correctness of answers are evaluated to see how the proposed composition procedure improves the reasoning quality.

4.3.1 Coverage

The first part of experiment looks into the coverage of reasoning results. The coverage is the ability a reasoning approach has to answer the input queries. If a reasoning approach gives an answer to the input query, we marked it as a "hit." The coverage is thus defined as the hit rate to a set of input queries:

$$hit\ rate = \frac{\#\ of\ answered\ queries}{\#\ of\ input\ queries}$$

Figure 4.1 compares the hit rate of the three approaches.

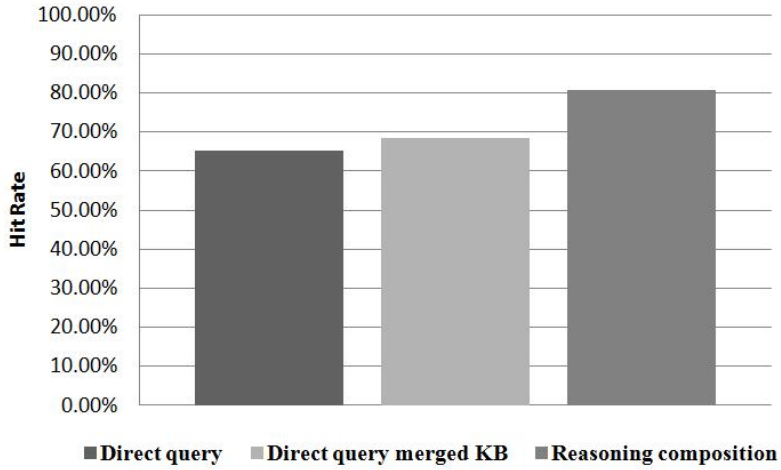


Figure 4.1: Hit rate of queries

The ConceptNet itself answered 1,961 queries. If we only asked the merged KB

of ConceptNet and WordNet without further reasoning, it answered 2,076 queries. The hit rate grew from 65.34% to 69.20% by combining the KBs. However, the hit rate of the merged KB is still not ideal for user agents to use. Our proposed reasoning composition approach answered 2,474 queries in this experiment (i.e. hit rate = 82.47%). It is shown that the reasoning composition improved the coverage of answers over 12% against the merged KB. Since the coverage represents the percentage of use's action that can be processed by the user agent, we think user agent can handle most actions with our proposed reasoning composition. The other unprocessed user actions can then be covered by the "fail-soft" design.

4.3.2 Correctness of Reasoning Results

In addition to the input queries answered by the merged KB, the reasoning composition approach answered other 398 queries. For the queries that were answered by merged KB and reasoning composition, we have high confidence on their accuracy since they have direct links to the input concept in the commonsense semantic networks. For other queries that are only answered by reasoning composition, they are shuffled and put on Amazon Mechanical Turk to be voted. Each reasoning result is rated as either relevant concept or non-relevant concept by at least 3 randomly selected workers. It is treated as a good answer if at least half of the workers rated it as relevant. Otherwise, it is considered as bad answer. There are 331 out of the 398 answers are rated as good answers (i.e. accuracy = 83.17%).

In our current implementation, we found that the bad answers are usually the answers that are too general or in wrong concept types. The first issue comes from

the generalizer and similar concept finder. For example, we got "mall", "school", and "beach" as the related location of "play soccer" because "play soccer" can be generalized to "play" and retrieve concepts that are places for playing. This problem suggests that the reasoning agents should selectively return the result to produce more accurate composition. The second issue results from the ambiguity of natural language concepts. Since concepts in ConceptNet are collected in natural language texts, the users often input texts that can forms sentences rather than the texts in the expected type. Therefore, we may get activities "buy product" and "spend money" as the answer to the query "related objects of smoker".

Both issues found from bad answers show the limitation in applying classical reasoning on collaboratively-built KBs. Since the data is noisy and represented in a flat network, there are exceptions in combining reasoning agents by planning. If we remove the bad answers from the reasoning results of reasoning composition, the hit rate still maintains at 80.23%. Our result demonstrates that it is still feasible to answer queries by composition of reasoning methods even if there are some constraints in applying logical reasoning on noisy KBs.

Comparison with Spreading Activation

The reasoning results are also evaluated against the spreading activation on English ConceptNet. Since the spreading activation cannot not differentiate all the types used in the 3,000 queries, we grouped actor and object into one type. The 398 queries that are answered by reasoning composition are grouped into 338 queries and asked spreading activation for answers. The spreading activation answered 167 out of the

338 queries. These answers were also put on Amazon Mechanical Turk to verify their correctness. We then mapped the good answer to 1 point and bad answer to 0 point. The average score for spreading activation was only 0.419; the reasoning composition got 0.916 for the same queries. Comparing the scores of spreading activation with our reasoning composition using a paired t-test showed that the average 0.497 point difference was statistically significant at the $p < 0.0001$ level, with a 95% confidence interval of (0.422, 0.572). We conclude from the comparison that the reasoning composition uses planning to combine reasoning agents gives it more flexibility and inferential power over the previous spreading activation approach.

4.4 Improving Coverage of KB

In order to evaluate the proposed approach to resource-bounded crowdsourcing of commonsense knowledge, we performed experiments using the ConceptNet APIs for accessing and contributing data.

4.4.1 Acquisition method: Virtual Pets

The questions generated by algorithm 6 were added to the Virtual Pets [15], a community-based game for collecting Chinese common sense. The players in Virtual Pets answer questions such as "*Spoon* is used for ___?"

The players in Virtual Pets are the agents defined in knowledge acquisition process. They may ask/answer their pets questions, vote good or bad to an answer they get, or use the commonsense points to exchange food for their pets. In aver-

age, there were 55 active players per day and each of them answers 3.16 questions in one day. Therefore, the resource bound Θ in Virtual Pets is the number of questions that can be answered by the users in one day, i.e. $\Theta \approx 55 \times 3.16 = 173.8$. Since the collaboratively-built knowledge base requires answers verified by consensus (i.e. duplicated answers), the actual resource bound Θ is smaller than 173.8 per day. If we try every possible question, it may take many years to get one answer for each question.

In order not to interfere in the interaction flow of the game, we created a virtual player to ask the generated questions to other players. We randomly selected 1/3 of the active players every day to ask them the generated questions from the virtual player. If the players think the question they are asked does not make sense, they can simply pass it and report it to the system. For different players, they may get same questions in the same day so that we can get the concensus answers from these players.

4.4.2 Experimental Result

In this experiment, we asked 480 questions in Virtual Pets and took 6 weeks to collect answers to these questions. Both the generated questions and collected answers were verified by players to evaluate their qualities. We also put the collected sentences to the Chinese ConceptNet to see how the generated questions improve the concept coverage within the resource limitation.

Concept Coverage

The first part of experiments looks into the coverage score of each concept in the Chinese ConceptNet before we conducted the proposed question generation process. There are 399 concepts whose coverage score ≥ 0.5 and 8,634 concepts whose coverage score < 0.5. Since the coverage score indicates the differences of inference results between Chinese and English ConceptNet, the result shows that the inference results of over 95% of concepts in Chinese ConceptNet were different from their mapped concepts in English ConceptNet.

Table 4.2: Example of concepts with different coverage scores

coverage score >= 0.5	coverage score < 0.5
動物 (animal)	華盛頓 (Washington)
建築物 (building)	經濟學 (economics)
食物 (food)	吸血鬼 (vampire)
房間 (room)	星座 (constellation)
空氣 (air)	巧克力 (chocolate)

When we compared the concepts with different coverage scores (see examples in table 4.2), we can easily find that most concepts with high coverage scores are generic concepts such as food and animal, whereas the concepts with low coverage scores are specific and culture-dependant concepts such as chocolate and vampire. We think it is the game design make players in a game tend to create easy and generic questions. If the questions are easy, they can get game points within a short period of time. With English ConceptNet as a guiding KB, it is much easier to create specific questions to ask players.

Quality of the Generated Questions

In the experiment, the question generation algorithm was used to produce 38,285 questions, with only 3,743 of which already in the Virtual Pets. This suggests that the proposed approach identifies mostly new features rather than existing features in the knowledge base for question generation.

Given the resource bound, 480 questions generated from the lowest coverage scores are selected to ask players. These questions are mixed with the questions generated by human players. We recorded the bad questions players reported in the six weeks and make a comparison between the two question sets in table 4.3. The percentage of good questions generated by our algorithm is 94.17% which is comparable to the ones generated by human players

Table 4.3: Quality of the generated questions

	By our algorithm	By players
# of questions	480	4,329
# of bad questions	28	264
% of good questions	94.17%	93.90%

Quality of the Acquired Sentences

Within the resource bound, i.e. $\Theta \approx 480$ in 6 weeks, we collected 3,788 distinct sentences. All collected sentences were shuffled to be voted. Each sentence is rated as either good or bad by 3 randomly selected players, and it is treated as a good sentence if two or more players rated the sentence as good. Otherwise, it is considered as a bad sentence. There are 3,249 out of the 3,788 sentences are rated as good

sentences by players (precision = 85.77%). Compared to the answers to player-generated questions (80% for answer count ≥ 2 [15]), we think questions generated by our algorithm are able to draw more good answers than the questions created by human players. In addition, all of these collected sentences are new to the Chinese ConceptNet since we generated questions from concepts with low coverage scores. Therefore, the collected sentences have more expected contribution to the knowledge base than the sentences collected from the original collection process.

Coverage Improvement

Table 4.4 summarizes the improvement in concept coverage from week 5 to week 8 in the proposed crowdsourcing process guided by the English ConceptNet. Taking week 0 as the baseline, the improvement is measured in terms of the number of concepts whose coverage score is < 0.5, denoted by $|c^-|$, and the corresponding percentage of improvement, denoted by Δ. Concept coverage improves by 33.51% at week 5, and steadily improved to a 37.02% increase by the 8th week.

Table 4.4: Improvements in concept coverage

	week 0	week 5	week 6	week 7	week 8
$\|c^-\|$	8,630	5,783	5,495	5,450	5,435
Δ	–	33.51%	36.33%	36.85%	37.02%

In addition to improving the original concepts in Chinese ConceptNet, 402 new concepts were introduced to the knowledge base. The statistics show that we can incrementally add new question-answer pairs so that the target knowledge base will approximate the guiding KB. However, it is interesting to observe that someconcepts

with low coverage scores may remain due to culturaldifferences and insufficient data in the guiding KB.

Chapter 5

Case Study

In order to have user agents equipped with the reasoning results easily, we developed REST API for user agent to access the reasoning results output by our integration system. An agent simply sends input query via the API to get the contextual information of the target concept. The reasoning function of two interface agents are replaced by our system to see their improvement in providing scenarios for video editing and dialog assistance interface.

5.1 Study 1: Storied Navigation

Storied Navigation [37] is a video editing system that helps editors compose a video story by selecting videos from a set of annotated videos. After each clip is imported into the system, Storied Navigation extracts the key concepts from the sentence annotation of clip and represents a sentence in a concept representation of contextually

related concepts. Editors can then type sentences in natural language to retrieve video sequences with similar concept representation.

In this study, we choose "the earthquake that struck Japan on March 11th, 2011" as our story domain. The corpus of videos is consisted of 23 clips with English descriptions collected from YouTube[1]. The description of clips served as the text annotation of the clip. In our annotation phase, these videos are processed by spreading activation and our reasoning API respectively to create concept representation.

From our record, Storied Navigation used 42 input queries during this annotation phase. There are 13 out of the 42 queries are answered by spreading activation with ConceptNet; 23 of them are answered by our system. Comparing the found actors and objects by the two approaches (see table 5.1), we can find that the spreading activation returns large amount but diverse answers, which require filtering before the interface agent use them. Using our system, the interface agent is able to handle more actions by being equipped with more relevant knowledge.

The quality of reasoning results brought the different feedbacks when an editor try to find a public speech of a leader by typing "a leader says...". As shown in figure 5.1, Storied Navigation with our system can find the video in which President Obama gave a speech because it includes "leader" as part of the concept representation of the retrieved video. On the other hand, Storied Navigation with spreading activation fail to find a relevant video because it brings many irrelevant concepts and none of them can be included into the concept representation of the annotated sentence

[1]http://www.youtube.com

Spreading Activation	Reasoning Composition
(measure distance, 0.2194), (president of united state, 0.2038), (event, 0.1638), (united state america, 0.1618), (government, 0.1511), (idea, 0.1511), (plant, 0.1477), (racist, 0.1307), (jail, 0.1307), (brat, 0.1307), (star movie, 0.1306), (famous person, 0.1306), (michigan, 0.1265), (american president, 0.1286), (chess piece, 0.1279), (fool, 0.1183), (gentleman, 0.1091), (idiot, 0.032)	(leader, 0.24), (politician, 0.2), (elect official, 0.12), (chief of state, 0.12), (coporate executive, 0.23), (academic administrator, 0.12), (elect leader, 0.12), (presidency, 0.12), (head of state, 0.12)

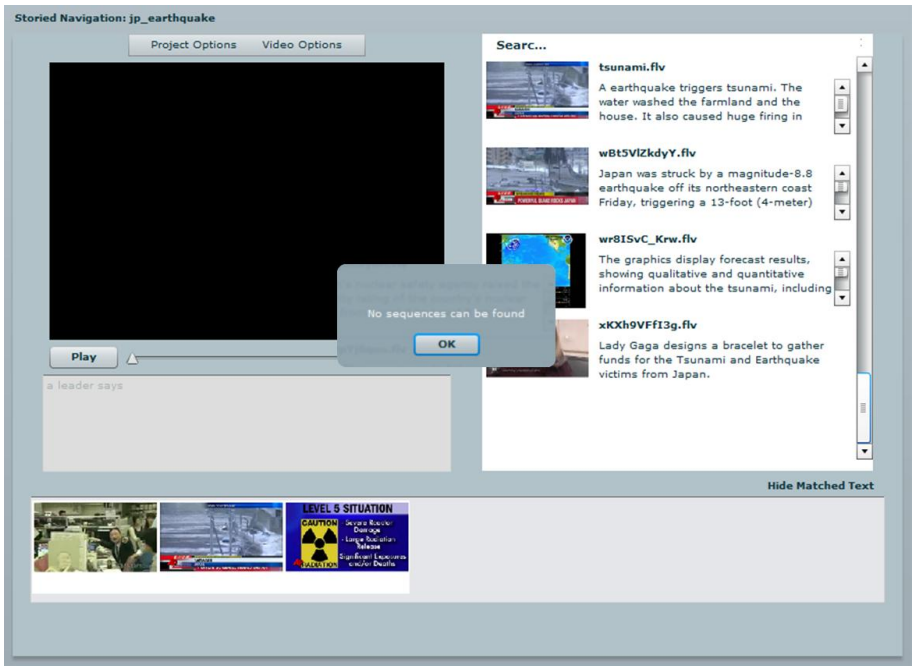
Table 5.1: Found related actors and objects of "president"

so that the similarity score is low. If an interface agent always fails to provide suggestion to user, it will be a great distraction to users.

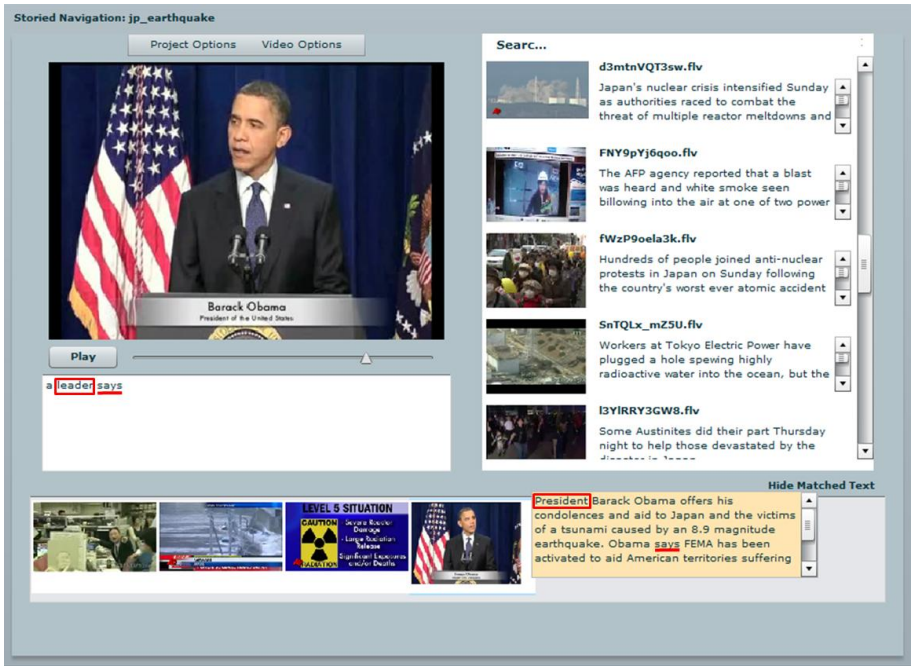
5.2 Study 2: Dialog Assistant

Dialog Assistant is another interface that helps users continue their conversation with others by associating the related items of a topic. It uses picture cards as communication cues. Since a conversation is similar to storytelling process, these picture cards are grouped into the following story elements: actor, activity/event, location, and object, which are correspond to our defined concept types. The agent generates the picture cue cards using the topic or the picture user select to suggest the plausible items for conversation. The Dialog Assistant helps user brainstorm about what should be include in their conversation.

We developed two versions of Dialog Assistant. One is directly querying the



(a) Storied Navigation with spreading activation



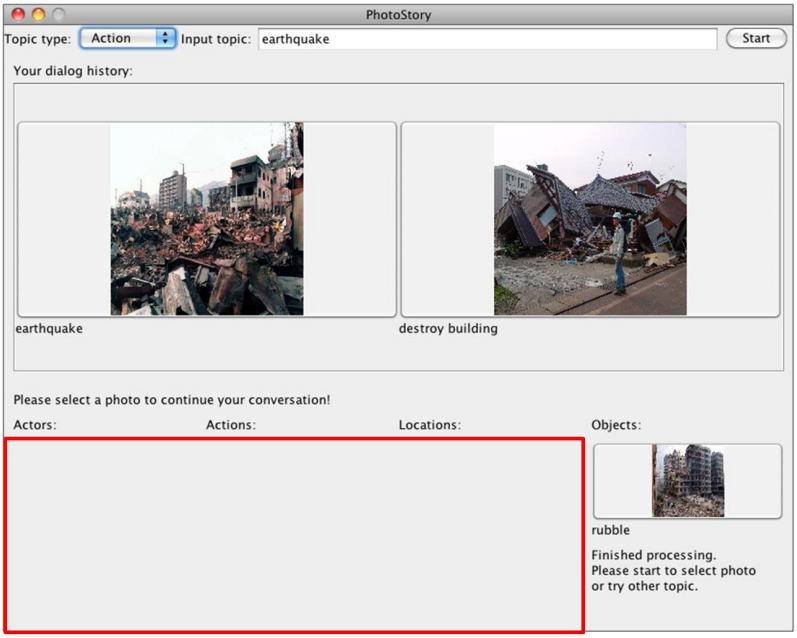
(b) Storied Navigation with our system

Figure 5.1: Storied Navigation suggests video when user types “a leader says...”

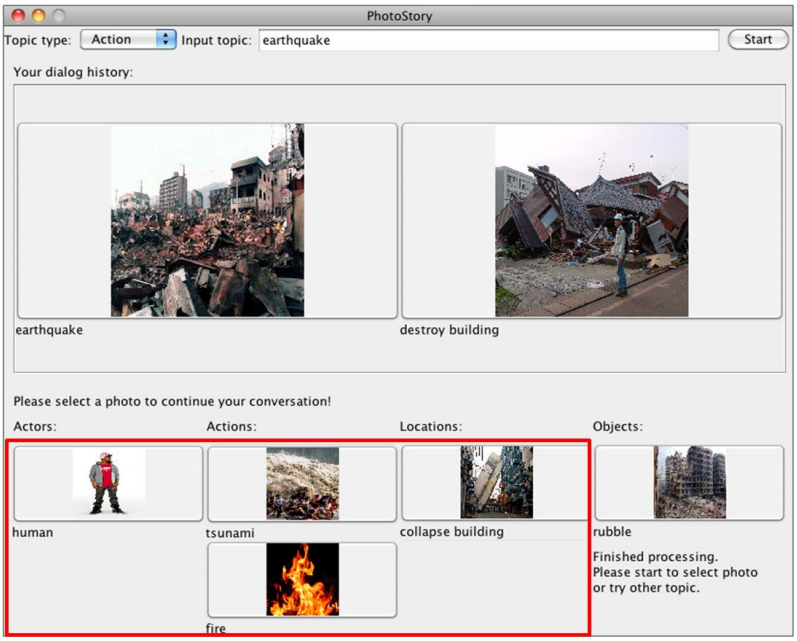
English ConceptNet for related items. Another is using our system to get the four kinds of story elements. Figure 5.2 shows their difference in handling the same action. Interface agent with our system provides more picture cues than the agent with direct query. Therefore, it is possible for users to get more associations with other items using the additional cues.

The selected picture cues in a conversation session forms an association sequence to a topic. Two hypotheses are to be verified: 1) "our system helps user produce longer association sequences" and 2) "these sequences are also reasonable to other people." A user was asked to compose 10 association sequences for the topic of "earthquake" and "travel." Each sequence was required to be consisted of 10 picture cues because a satisfactory conversation should keep a certain length. Unfortunately, the user can only produce association sequence of 2 or 3 picture cues using the Dialog Assistant with direct query. The user successfully compose 6 sequences for "earthquake" and 4 sequences for "travel" using the Dialog Assistant with our system. After creating the association sequences, 35 online users were recruited to rate each association pair in these sequences. If the two concepts in a pair are highly associated and related the topic the pair got 5 points. Otherwise, it got 1 point. Figure 5.3 shows the numbmer of pairs for each medium score.

There are 55 pairs (medium score = 4 or 5) out of 90 pairs are considered reasonable and related to topic. 18 pairs (medium score = 2 or 1) are considered being composed of less relevant concepts and not so related to the specified topics. Most of the cues suggested by Dialog Assistant form a reasonable transition of story elements to other people. However, we also observed individual differences in



(a) Dialog Assistant with direct query



(b) Dialog Assistant with our system

Figure 5.2: Dialog Assistant gives picture cues when user selects “destroy building”

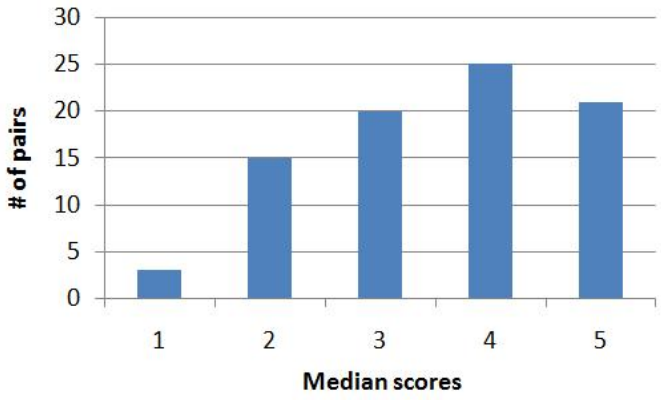


Figure 5.3: Number of pairs for each medium score.

associating multiple concepts. The commonsense knowledge may plays a role to fill in the gaps.

Chapter 6

Conclusion

This chapter sumarizes the contribution of this thesis and provides future work to improve our commonsense knowledge integration system.

6.1 Summary of Contribution

This thesis proposes a multiagent system to integrate commonsense knowledge. Instead of merging multiple KBs into a single one, we combine reasoning methods to answer queries. Three key mechanisms are proposed to achieve matchmaking and interoperation of KBs. By matching KBs that contain the targeted knowledge, leveraging multiple reasoning methods, and improving coverage of KB through a guiding KB, we are able to answer more queries with higher accuracy.

Experiments have been conducted with ConceptNet, WordNet, and Wikipedia to verify the feasibility of the mechanisms. The first experiment compared the

matchmaking results with user-produced ranking lists. The matchmaking using our proposed capability model showed accuracy of over 85% and a positive rank coefficient. With the capability model of KBs, the system can match the most relevent reasoning agents to applications so that complex reasoning tasks can then be built on top of the system. The sencond experiments compared the reasoning results of reasoning composition, directly querying merged KB, and spreading activation. The results showed 10% improvement in hit rate over direct querying the merged KB, and 40% improvement in correctness over the conventional spreading activation. The third experiment generated questions for growing commonsense KBs, such as the Chinese ConceptNet collected via Virtual Pets, using English ConceptNet as a guide. The results showed that the generated questions and their collected sentences are as good as the ones collected from original acquisition process (precision = 94.17% for questions and 85.77% for answers) with an improvement in concept coverage by 37.02%. The proposed approach can be embedded into a perpetual crowdsourcing process to improve the coverage of KBs. It is also easy to target specific domains in acquisition of commonsense knowledge within the resource bound.

Finally, two case studies on video editing interface and dialog assistance interface demonstrated this proposed multiagent reasoning system is easy to be embedded into interface agents so that they can perform robustly.

6.2 Future Work

The current system presents an initial step towards integration of commonsense KBs. This thesis focuses on providing integrated contextual reasoning with commonsense semantic networks for interface agents. In order to bring applications easier ways to utilize the commonsense reasoning results, we may extend the capability of reasoning agents to other types of reasoning methods. For example, we may include temperal reasoning into one of our reasoning agents using the temperal knowledge in Cyc. In addition, more in/output format and composition rule may also need to be included to serve more kinds of reasoning requests. Our proposed multiagent system is flexible to incorporate more reasoning methods to enhance the system's capability in providing useful reasoning results to applications.

Bibliography

[1] E. Cambria, A. Hussain, C. Havasi, and C. Eckl. AffectiveSpace: Blending common sense and affective knowledge to perform emotive reasoning. In *Proceedings of the CAEPIA Workshop on Opinion Mining and Sentiment Analysis*, 2009.

[2] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010.

[3] T. Chklovski. Learner: a system for acquiring commonsense knowledge by analogy. In *K-CAP ' 03: Proceedings of the 2nd International Conference on Knowledge Capture*, 2003.

[4] T. Chklovski and Y. Gil. An analysis of knowledge collected from volunteer contributors. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[6] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[7] I. Eslick. Searching for commonsense. Master's thesis, Massachusetts Institute of Technology, 2006.

[8] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Methods for domain-independent information extraction from the web: an experimental comparison. In *Proceedings of the 19th national conference on Artifical intelligence*, 2004.

[9] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[10] C. Havasi, J. Pustejovsky, R. Speer, and H. Lieberman. Digital intuition: Applying common sense using dimensionality reduction. *IEEE Intelligent Systems*, 24(4):24–35, July 2009.

[11] C. Havasi, R. Speer, and J. Alonso. ConceptNet 3: A flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September 2007.

[12] C. Havasi, R. Speer, and J. Pustejovsky. Coarse word-sense disambiguation using common sense. In *2010 AAAI Fall Symposium on Common Sense Knowledge (FSS10)*, 2010.

[13] Y.-L. Kuo and J. Y.-j. Hsu. Bridging common sense knowledge bases with analogy by graph similarity. In *2010 AAAI Workshop on Collaboratively-Built Knowledge Sources and Artificial Intelligence*. AAAI Press, July 2010.

[14] Y.-L. Kuo and J. Y.-j. Hsu. Goal-oriented knowledge collection. In *2010 AAAI Fall Symposium on Common Sense Knowledge (FSS10)*, 2010.

[15] Y. L. Kuo, J. C. Lee, K. Y. Chiang, R. Wang, E. Shen, C. W. Chan, and J. Y.-j. Hsu. Community-based game design: experiments on social games for commonsense data

collection. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, 2009.

[16] D. B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[17] H. Lieberman. User interface goals, AI opportunities. *AI Magazine*, 30:16–23, 2009.

[18] H. Lieberman, H. Liu, P. Singh, and B. Barry. Beating common sense into interactive applications. *AI Magazine*, 25:63–76, 2004.

[19] H. Liu, H. Lieberman, and T. Selker. GOOSE: a goal-oriented search engine with commonsense. In *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002*, 2002.

[20] H. Liu and P. Singh. ConceptNet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226, 2004.

[21] D. Martin, M. Burstein, E. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services. Technical report, 2004.

[22] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimaera ontology environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 2000.

[23] S. McIlraith and T. C. Son. Adapting golog for composition of semantic web services. In *Proceedings of the 8th International Conference on Knowledge Representation and Reasoning*, Toulouse, France, April 2002.

[24] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journel*, 12(4):43–54, November 2003.

[25] G. A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[26] M. Minsky. *The Society of Mind*. Simon and Schuster, 1988.

[27] A. Newell and G. Ernst. The search for generality. In *Proceedings of IFIP Congress*, 1965.

[28] M. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, and A. Unruh. Active information gathering in InfoSleuth. *International Journal of Cooperative Information Systems*, 9(1/2):3–28, 2000.

[29] N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 2000.

[30] Organization for the Advancement of Structured Information Standards (OASIS). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, 2007.

[31] K. Panton, C. Matuszek, D. Lenat, D. Schneider, M. Witbrock, N. Siegel, and B. Shepard. Common sense reasoning – from Cyc to intelligent assistant. *Lecture Notes in Computer Science*, 3864:1–31, 2006.

[32] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity - measuring the relatedness of concepts. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, 2004.

[33] A. Preece, K. Hui, A. Gray, T. Bench-capon, D. Joes, and Z. Cui. The KRAFT architecture for knowledge fusion and transformation. In *Proceedings of the 19th SGES International Conference on Knowledge-Based Systems and Applied Artificial Intelligence*, 1999.

[34] J. Rao, P. Kungas, and M. Matskin. Logic-based web services composition: from service description to process model. In *Proceedings of the 2004 International Conference on Web Services*, pages 446 – 453, July 2004.

[35] J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. *Semantic Web Services and Web Process Composition*, 3387:43–54, 2005.

[36] L. Schubert and M. Tong. Extracting and evaluating general world knowledge from the brown corpus. In *Proceedings of the HLT-NAACL Workshop on Text Meaning*, 2003.

[37] E. Y.-T. Shen, H. Lieberman, and G. Davenport. What's next?: emergent storytelling from video collection. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, CHI '09, pages 809–818, New York, NY, USA, 2009. ACM.

[38] M. P. Singh and M. N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.

[39] P. Singh. The public acquisition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium*, 2002.

[40] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems, 2002 - DOA/Coop IS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, 2002.

[41] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377 – 396, 2004.

[42] A. Smirnov, M. Pashkin, N. Chilov, and T. Levashova. Multi-agent architecture for knowledge fusion from distributed sources. *Lecture Notes in Artificial Intelligence*, 9(2296):293–302, 2002.

[43] R. Speer, C. Havasi, and H. Lieberman. AnalogySpace: Reducing the dimensionality of common sense knowledge. In *Proceedings of AAAI-2008*, 2008.

[44] M. Strube and S. P. Ponzetto. WikiRelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[45] G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In *Prceedings of the 7th International Conference on Artificial Intelligence*, 2001.

[46] L. von Ahn, M. Kedia, and M. Blum. Verbosity: A game for collecting common-sense knowledge. In *ACM Conference on Human Factors in Computing Systems (CHI Notes)*, pages 75–78, 2006.