2020

Identifying Knowledge Gaps Using a Graph-based Knowledge Representation

Daniel P. Schmidt
Wright State University

IDENTIFYING KNOWLEDGE GAPS USING A GRAPH-BASED KNOWLEDGE REPRESENTATION

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

by

DANIEL P. SCHMIDT
B.S.C.S., Wright State University, 2019

2020
Wright State University

Wright State University
GRADUATE SCHOOL

April 10, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Daniel P. Schmidt ENTITLED Identifying Knowledge Gaps Using a Graph-based Knowledge Representation BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Pascal Hitzler, Ph.D.
Thesis Director

Mateen Rizki, Ph.D.
Chair, Department of
Computer Science and Engineering

Committee on
Final Examination

Pascal Hitzler, Ph.D.

Michael Raymer, Ph.D.

John Gallagher, Ph.D.

Christopher Myers, Ph.D.

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

ABSTRACT

Schmidt, Daniel P. M.S., Department of Computer Science and Engineering, Wright State University, 2020. *Identifying Knowledge Gaps Using a Graph-based Knowledge Representation.*

Knowledge integration and knowledge bases are becoming more and more prevalent in the systems we use every day. When developing these knowledge bases, it is important to ensure the correctness of the information upon entry, as well as allow queries of all sorts; for this, understanding where the gaps in knowledge can arise is critical. This thesis proposes a descriptive taxonomy of knowledge gaps, along with a framework for automated detection and resolution of some of those gaps. Additionally, the effectiveness of this framework is evaluated in terms of successful responses to queries on a knowledge base constructed from a prepared set of instructions.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to thank my advisor, Dr. Pascal Hitzler, for starting me down this path, as well as always guiding and encouraging me to keep improving myself. I am forever grateful for his mentorship during my time at Wright State.

I also want to thank my Air Force Research Laboratory (AFRL) project sponsor, Dr. Chris Myers, for granting me the opportunity to research this topic and for introducing me to the world of cognitive science.

Finally, a huge thank you to my parents for their constant love and support along this journey, as well as for the many late-night conversations helping to keep my head above water.

This work was, in parts, supported by AFRL, the Southwestern Ohio Council for Higher Education, and Wright State University through award RH4-WSU-19-2 of the Defense Associated Graduate Student Innovators program, titled Machine Reasoning to Identify Inconsistent Knowledge. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of AFRL, the Southwestern Ohio Council for Higher Education, or Wright State University.

To Tiffany.

You are my motivation and the love of my life.

1

Introduction

A knowledge gap arises when there is limited or missing information or capabilities and typically causes an agent to produce inefficient or erroneous behavior, even leading to a process halting. In a typical knowledge-based system, if a question is asked that requires knowledge which the system does not have (whether external vocabulary, situationally relevant knowledge, information that connects two seemingly disjoint pieces of information, etc.), then the system simply returns an error or does the equivalent of saying "I don't know", without further feedback. This event is an example of a knowledge gap in action.

Knowledge gap identification in knowledge base systems is an important, emerging concept. Its importance stems from the fact that automated knowledge acquisition is playing an increasingly crucial role in artificial intelligence and synthetic assistant technology, but knowledge gap identification has yet to be adequately addressed. As another example where knowledge gap identification and resolution can be of service, question answering systems (e.g. Wolfram Alpha [22]) are by nature designed to take in questions from users regarding an existing knowledge base and attempt to return accurate answers to the questioner; correctly ingesting knowledge is critical to such systems. For this reason, it is important to not only identify gaps in knowledge, but then find ways to resolve these gaps. Some examples of knowledge gap identification, as well as gap resolution, are provided in

this thesis.

Two major paths have been identified that may be taken to improve results from such systems. One path would be to extend the information to which the system has access, whether by storing more information internally or having external sources that the system can query to augment its knowledge base when it receives a question that causes a gap in the system's knowledge. However, in the event of failing to find the relevant information, this system will likely still return a flat "Unknown" response. Another approach that would not only lead to less "I don't know" answers, but would make it easier to resolve such answers when they do arise, would be to increase the depth of processing over the integrated knowledge to improve the potential for responses. For example, if a system receives some information like "Anakin Skywalker is Luke Skywalker's father. Darth Vader told Luke Skywalker 'I am your father'", a fairly shallow question could be "Is Darth Vader Luke Skywalker's father?", while a deeper question which could show improvement if answered correctly could be "Is Darth Vader Anakin Skywalker?". To this end, automated identification and resolution of gaps, to the extent possible, appears to be a step in the right direction when it comes to increasing a system's depth of processing, as well as a mechanism toward extending the system's available knowledge and application of knowledge.

With automated knowledge gap identification and resolution, it becomes possible for a system to operate using both a form of "passive" and "active" gap identification at the same time - "passive" gap identification refers to gap identification that occurs prior to user querying, while "active" identification is triggered by a user query revealing or causing a gap. In its passive form, the system receives its initial set of instructions/information in a first pass, then automatically checks this knowledge for any gaps which it can identify. Passive identification allows for the user/teacher of the system to be immediately notified of any potential problems without needing to ask questions or wait for the error to be encountered later on; this could be seen as equivalent to compile-time error checking. In the active form of checking, gaps are detected when a user enters a query into the system.

The query could either lead to detection of a gap which already exists in the knowledge base or a gap could arise due to a disconnect between the query and the existing information in the knowledge base; in either scenario, the gap will be identified. Active knowledge gap identification allows the system's user to be quickly and accurately informed as to what has gone wrong in the event of an error in knowledge retrieval, thus leading to better results and faster recovery time; this could be likened to run-time error checking.

With regards to knowledge gap resolution, there are two primary ways in which the identified gaps can then be handled. First, a gap can be automatically resolved due to the system knowing the likely causes of the type of gap identified and having a method in place to attempt resolution of such a gap without needing user intervention. Alternatively, in the instance where the system does not have a method in place to resolve the gap or the automated resolution fails to solve the problem, the system can more accurately inform the user as to why the gap arose, explaining which gap has occurred and what may be needed from the user in order to resolve it. In the first case, the user may or may not even be informed of the gap, but operation continues smoothly. In the second case, it is much easier for the user to understand what happened and why the gap arose, as there is context given, and debugging the knowledge base or query likely becomes a simpler task.

The hypotheses, therefore, are twofold.

- First, that having automated knowledge gap identification and resolution can provide a higher percentage of non-"Unknown" responses to queries than a system which lacks such a mechanism.

- Second, from a more qualitative perspective, that having gap identification, even if there is no automated resolution available, provides insight into where a problem arose with regards to answers which returned "Unknown", enabling users to more quickly devise a new query or debug a problem with the knowledge base than simply receiving an error without explanation.

To evaluate these hypotheses, a system has been developed which can ingest a set of instructions, generate a knowledge graph, and be queried; this system is named KOIOS, after the Greek Titan of intelligence. KOIOS detects knowledge gaps as they are encountered, whether due to a query ("actively") or independent of human input ("passively"); it then endeavors to resolve the gaps through a number of different resolution methods to be described in this thesis.

Chapter Overview

The rest of the thesis is organized as follows.

Chapter 2: Preliminaries introduces important preliminary information. This includes information about the tasks used for testing and evaluation of the KOIOS system, as well as background information about the restricted form of English used for input into KOIOS.

Chapter 3: Related Work covers other research which explores similar topics and goals as this thesis. Specifically, works are considered which deal with general knowledge gap detection, Visual Question Answering, and Schema/Ontology Alignment.

Chapter 4: Research Contributions contains a high-level taxonomy of knowledge gaps as well as a high-level explanation of each of the working parts within the KOIOS system. This taxonomy was not originally within the scope of the research, but proved to be a major development, as it appears to be a novel summary of information in addition to serving to guide the goals of KOIOS development. The section on the KOIOS system also explains how the system identifies and resolves different types of knowledge gaps in order to improve the performance of knowledge base queries.

Chapter 5: Evaluation explains the design and results of the evaluation.

2

Preliminaries

To develop a system which could construct a knowledge base from instructions, instruction sets were needed which could be used as the source of knowledge. Additionally, in order to minimize out-of-scope work, a mechanism was needed to circumvent the difficulty of natural language processing. As the introduction states, this section is intended to present important background information regarding these items. Section 2.1 explains some background about the Psychomotor-Vigilance Task (PVT) and Visual Search Task (VST), which are used as proof-of-concept knowledge sets. Section 2.2 gives information about the Attempto Project and its derivations, which are used to minimize the complexity of language processing.

2.1 Instruction Sets

This section describes the two cognitive science tasks used as proof-of-concept tests; the basic instruction sets for these tasks were provided by the research sponsor at the Air Force Research Laboratory (AFRL). Each task's subsection will briefly cover what the actual task description is for the task, why the task was selected, and and some background information on the task.

2.1.1 Psychomotor-Vigilance Task

The Psychomotor-Vigilance Task (or PVT) is a system proposed in 1985 [15], designed to evaluate a subject's vigilance during relatively repetitive and menial tasks over sustained periods of time. This task has been used to develop computational models of fatigue over time caused by sleep loss [47] and is quite simple, consisting essentially of a button next to a box in which a letter appears at random; when a letter appears in the box, the subject must click on the button. In the context in which the PVT is used in this research, the subject also must remember the letter, as it may feed into other tasks such as the Visual Search Task. This task was chosen as a good starting point for developing and testing the system, both in terms of instruction input handling as well as beginning to identify simple gaps, due to its simplicity while still retaining the opportunity for some simple gaps to arise.

2.1.2 Visual Search Task

The Visual Search Task (VST) [42] is a significantly more advanced task in comparison to the PVT. The task's goal is to examine speed of identification of a target within a field of distractors and analyze how different changes to the task (such as the number of distractors or the degree of similarity between the distractors and the target) affect eye movement and time to determine if a target is present or absent from the display. The specific layout of the Visual Search Task, as a standalone task, is still relatively simple - the subject is told to look for a target item, which is an object made up of a features (e.g., shape, color, rotation, etc.). This target appears on a canvas, surrounded by a field of distractors (containing the same shape in a different color and/or a different shape in the same color). The subject must attempt to find the target they were provided, at which point they press a "Present" button; if one does not find the target s/he responds by pressing a button labeled "Absent". Handling the VST added some additional challenges in comparison to PVT - notably the added complexity in its conditionals, such as the addition of a negative conditional ("If the

target is not found, press 'Absent'."). This added complexities in language constructions beyond those presented by the PVT as well as the potential for new knowledge gaps. Overall, VST works well as a second-order problem, generally being a more complex task with potential for more gaps.

2.2 Attempto Project

In this section, the primary elements of the Attempto Project used as a surrogate for natural language processing in the system are introduced. Additionally, the reasoning behind why these elements were chosen is outlined, and an explanation of how they are directly related to each other and the task at hand is given.

2.2.1 Attempto Controlled English

Attempto Controlled English (ACE; [16]) reduces the scope of English constructions so as to easily translate it into something which is immediately translatable to first-order logic. While the first-order logic properties of ACE are not of direct interest, this reduction in complexity helps to avoid the difficulty of natural language processing for instruction and query inputs. Consequently, ACE is used as the first step in translating the natural language instruction sets into a knowledge base. Using ACE limits not only the vocabulary of the inputs, but also the grammar. In terms of vocabulary, ACE is constrained to content words (major parts of speech such as nouns, verbs, adjectives, and adverbs), a select subset of additional function words, such as certain conjunctions and prepositions, and phrases, such as "there is" or other such commonly used phrases with clear meaning. These constraints mean that most, if not all, phrases in ACE are relatively short and simple, as well as highly disambiguated. Because of this, ACE can be easily translated into different representational forms including Discourse Representation Structures (DRS), allowing for easier use

of the incoming language in an automated system, especially as ACE can be easily and quickly converted into DRS through the Attempto Parsing Engine (APE), as explained in the following subsections - the DRS derived from ACE are particularly useful in allowing an automated system to process inputs.

2.2.2 Attempto Parsing Engine

The Attempto Parsing Engine (APE; [16]) contains a default general lexicon of content words and function words and can be expanded by user-provided, domain-specific lexicons of content words. One notable feature is that users can also provide words not found in any of APE's available lexicons but still indicate the part of speech by a prefix, allowing for flexibility in the ACE input, especially in the case of a "black box" scenario where the lexicon would be unknown. Usefully, APE is made public as a web service which is freely available (its source code can also be downloaded, modified as needed, and/or run locally).

APE allows for ACE translation into several different forms of output - several versions of syntax trees, paraphrased versions of the ACE, multiple different representations of the DRS which derive from the ACE, and even Web Ontology Language (or OWL) [33] derived from the DRS. The textual representation of the DRS generated from the ACE input eases the burden of language processing over the presented instructions.

2.2.3 Discourse Representation Structures

The syntax and structure of DRS [17] is derived from ACE via APE and is highly disambiguated. There are DRS atoms for nouns (or "objects"), verbs (or "predicates"), and other linguistic structures, such as negations, conditionals, and questions, to name a few. If a set of ACE phrases is converted to DRS, the identifying reference variables in the resulting atoms can also be referenced by future atoms, allowing for relationships between the different atoms to be explicitly identified. DRS provides a limited set of potential inputs,

eliminating the need to develop a part of speech tagger or language parser. With DRS, atoms are handled according to their tag. For example, object tags get processed in one consistent manner, predicate tags get handled with a limited amount of variability (there are a few "hardcoded" terms to deal with), and conditionals and negations are clearly identified. Consequently, it was easy to create a knowledge base and make the input processing portion highly generalizable, as any set of DRS instructions can be handled once there are processes in place to handle each of the structures that DRS offers.

3

Related Work

As previously addressed, published research into knowledge gap identification and resolution seems to be very few and far between. In this chapter, a number of papers are summarized which touch on the topic tangentially. While there does not seem to be any work that directly approaches the problem addressed herein of identifying specific gaps in knowledge, there are a number of fields which have to do with verification and alignment of information or rules. In Section 3.1, works that have to do with knowledge gap detection as a general concept are discussed. Section 3.2 considers the field of Visual Question Answering and what is being done there with regards to knowledge gaps. Finally, Section 3.3 looks at schema and ontology alignment at a cursory level as a form of knowledge gap detection.

3.1 General Knowledge Gap Detection

This section presents the works that are most similar to what this thesis is focused on - namely, they directly address identification and resolution of some form of knowledge gap; each of these works is tangentially aligned with this thesis's topic [37], [30].

3.1.1 Rule Verification: COVER

The most recent substantive research found in terms of rule verification is the COVER program found in [37], written in 1994. This work summarizes the major research in rule verification (also referred to as "anomaly detection") methods proposed over the period of 1982 to 1991 and performs an empirical study to check the performance and usefulness of these methods in terms of real-world knowledge bases. This research operates primarily in the realm of rule-based systems; the publication describes a knowledge base as a rule base and a declaration set, where a declaration set is made up of a set of goal literals, a set of input literals, and a set of semantic constraint expressions. The empirical study performed in the paper found that the anomaly detection principles worked in real-world scenarios and inexpensively indicated where there were anomalies in the knowledge base, thus allowing for corrections and improved performance. This study is encouraging, as it supports the theory that knowledge gap identification provides some benefit and insight into where errors arise.

However, the work in the COVER research does not quite align with what is proposed in this thesis, as the principles in COVER are specifically focused on logical correctness of a rule set. While this does address a certain type of knowledge gap (stemming from inconsistencies arising from conflicting rules, unnecessary conditions, or unreachable conclusions, to name a few), it only looks at the logical correctness of the rule set and only identifies gaps without resolving them. COVER's focus, on the other hand, is on logical rule sets, the focus of this thesis is to rather analyze a set of general instructions and look at more qualitative gaps.

3.1.2 Knowledge Gap Detection for Active Learning

There is also research centered on detecting knowledge gaps with regards to a robot engaged in active learning by classifying data [30]. This paper proposes a new strategy for an

active learning method which would reduce the necessary amount of training data by using automated knowledge gap detection and communicating the gaps which are discovered to the teacher. An interesting concept which this paper presents is the distinction between introspective and extrospective knowledge gap detection and communication.

In terms of knowledge gap detection, extrospective detection is defined as detecting knowledge gaps in relation to particular situations or problems which arise while training is ongoing. Introspective detection, on the other hand, is not linked to any specific task, but rather is triggered by an internal mechanism intended to detect knowledge gaps. When introspective detection is triggered, no real input is used, but rather the robot uses its current knowledge base and, in the words of the authors, "hallucinates" sensorial inputs; by their explanation, this "hallucination" occurs by sampling over distributions of feature values it uses. When it comes to knowledge gap communication, the extrospective case they describe is once again the simpler of the two - in communicating the gap, the robot references an existing instance as an example of the case in which the gap was encountered. Introspective gap communication consists of taking the feature values which make up the knowledge gap and mapping them back to action parameters to understand what would have caused the input that triggered the gap, then returning this information to the teacher.

Notably, this paper uses "knowledge gap" to refer to what is essentially uncertainty about the robot's classification of some set of data; the general scope of the research is essentially limited to active learning. Despite this, this paper can be considered to be related to this thesis as it does conceptually approach the same core issue - finding out where information is lacking or incorrect and resolving it. In this case, the exact flaw in the information is not necessarily able to be pinpointed, although introspective communication does allow for at least generating information on what the input that caused the problem is. The resolution process comes from simply communicating with the teacher and receiving needed information in return which allows the robot to resolve the gap. The introspective and extrospective aspects also have some light parallels to the way KOIOS handles gaps,

especially in terms of detection; extrospective detection could have similarities drawn with KOIOS' knowledge gap detection which occurs on user questions, while introspective detection could be compared to passive gap detection on the existing instruction base.

3.2 Visual Question Answering

Visual Question Answering (VQA) is an AI problem based around taking in an image and a natural language question about the image. Originally proposed in [31], VQA uses images with annotations to represent the items in the graph and their relationships to each other; with this information available, a VQA system will then parse natural language questions which are asked of it regarding the image and attempt to resolve the question based on the knowledge which it has available. As the name suggests, this is at its core a question answering problem, so there is certainly a place for knowledge gap identification and resolution here to answer the question more accurately. This is even more salient when one considers [2], which proposes a system to handle open-ended and free-form questions asked about images. This not only expands the number of possible questions and the potential difficulty of the questions, but also opens the door to multiple answers being correct; both of these elements increase the potential for knowledge gaps as there may be unclear elements which need to be bridged or new terms introduced through the free-form questions.

There is some work which does aim to resolve such knowledge gaps involving new terms or unknown information, as found in [48]. This proposes a way to integrate external knowledge contained in DBPedia [3] or some other such external knowledge base by searching for information regarding the major attributes identified in the image. The system requests information about the attributes from whichever knowledge base is used and then uses Doc2Vec [26] to extract meaningful semantic information about the attributes and craft a simple but correct answer to the question asked. This clearly demonstrates a knowl-

edge gap being identified and resolved (the gap being that external knowledge is required and the resolution being the automated retrieval and integration of such knowledge), and the experimentation performed by the paper's authors shows that this process significantly improves performance on questions which have to do with "common sense" or external knowledge, especially "Why" questions or questions that have to do with the purpose of an object. While this work implicitly handles knowledge gaps by its very nature, it does not inform the user of when a gap is identified, nor does it appear to adjust its behavior based on the type of gap found - rather, whenever a gap is identified, it always calls for the external knowledge and handles it in the same manner.

3.3 Schema and Ontology Alignment

Schemas and ontologies are both structures to represent and model data, with some differences with regards to how and to what extent semantic information is encoded into the model. Both schemas and ontologies may have instances when they need to be aligned; namely, when two distinct schemas or ontologies containing similar data need to be fused, there must be a matching made for which items are related and which data are distinct. Many different schema matching and ontology matching methods are discussed at varying levels in [45]. One interesting point made in the paper is that schemas do not have any explicit semantics encoded into the data, while ontologies do. Thus, a schema matching will by nature have one more type of knowledge gap which must be addressed when compared to an ontology matching; namely, resolving the meaning of the terms encoded in the schema. In a very general form, an alignment between two schemas/ontologies consists of a set of mapping elements. Each mapping element essentially gives a confidence value for a relation between an entity from the first schema/ontology and an entity from the second. In attempting to determine relations and confidences between entities, there is a wide variety of knowledge gaps which must be bridged; each of the matching techniques

surveyed in this paper addresses a knowledge gap in some way. However, none of them appear to explicitly alert the user to which gap was encountered and what was done to resolve it. Despite this, at their core, gap detection and ontology alignment share some common features.

4

Research Contributions

This chapter describes the complete survey of knowledge gaps developed over the course of this research, as well as a high-level overview of KOIOS, the Python system developed to automatically detect and resolve knowledge gaps. The taxonomy provided here is the most up to date at the time of this writing, but gaps may be added or removed in the future as work continues toward improving this research.

There are two major sections to the rest of this chapter. Section 4.1 contains a description of the broad categories of knowledge gaps identified and goes into some detail about each type of gap. Section 4.2 explains at a high level each of the major functions of the Python system, as well as which gaps it can presently address and how it addresses them.

4.1 Knowledge Gap Taxonomy

In the goal of better understanding what the system would need to be able to do, research began by attempting to understand and classify different types of "knowledge gaps". A knowledge gap arises when there is limited or missing information or capabilities, which causes an agent to produce inefficient or erroneous behavior, even potentially leading to a process halting. These gaps can cause incorrect responses as the recipient of the informa-

tion may attempt to fill in the blank by faulty inference or may simply fail to complete the task. It was determined that having a survey of the major types of knowledge gaps would be highly beneficial and a major step in understanding which gaps could be reasonably addressed in the limited timeframe available for this research. Searching for such a survey led to research into the realm of cognitive science and, more specifically, meta-cognition and begin development of a descriptive taxonomy of knowledge gaps, as such a taxonomy does not currently exist.

This search led to early work by Dedre Gentner and Allan Collins [18], in which the concept of a "lack-of-knowledge inference" was encountered; namely, the tendency to believe that the lack of knowledge about some suggested event or thing which would be important enough to commit to memory means that the proposed event or thing is untrue. Collins, Warnock, et al. [13] propose that this may be a safe assumption in a closed world, but becomes significantly more dangerous in an open world, where a better response would be to admit to not knowing. The authors alluded to the concept of knowledge gaps, but did not specifically cover the topic.

Radeka [38] proposed three major goals for resolving knowledge gaps. The first is to establish facts which are not already known, which is the primary goal of resolving gaps through the proposed system in this thesis. The second reason listed is to develop alternative options to existing solutions, as resolving a gap could create a new link which would reveal an alternate way of doing a task or answering a question. The third reason is to establish boundary conditions on a known problem, i.e. to limit the world.

Despite the resources listed above, no work was found directly categorizing types of knowledge gaps, so one was developed. The general causes of knowledge gaps relating to incoming information were considered and grouped into four overarching major categories. The taxonomy was pulled together from a number of existing papers to derive twenty different types of knowledge gaps (see Figure 4.1). While this taxonomy is not exhaustive and was initially focused on knowledge gaps relevant to the goal of this thesis, it does

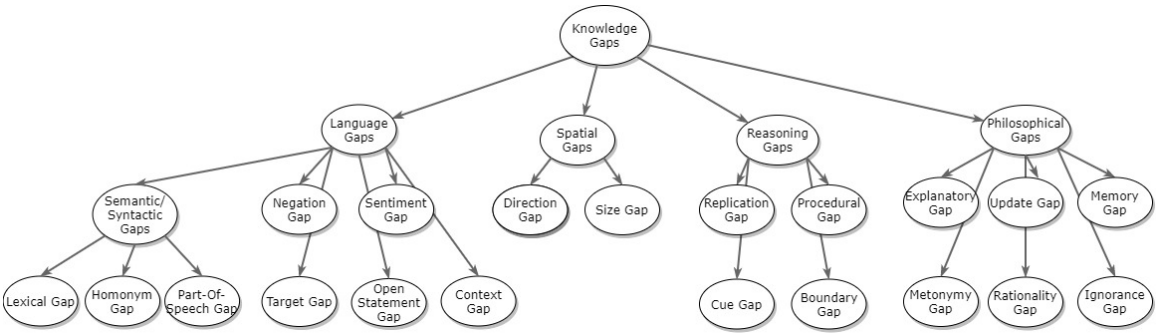

Figure 4.1: Hierarchy of Knowledge Gaps.

cover a number of other gaps which were encountered during the research. This taxonomy is covered in detail in the following sections.

Figure 4.1 shows a hierarchy of the gaps described below.

4.1.1 Language Gaps

Language Gaps, as the name suggests, are knowledge gaps which arise from disconnects in somebody's understanding of language. These typically manifest as misunderstandings in communication and thus instructions being ignored or incorrectly performed.

4.1.1.1 Semantic and Syntactic Gaps

4.1.1.1.1 Lexical Gap A Lexical Gap is a simple concept - when an agent receives a term that is not in their existing vocabulary, whether that be a word in another language or a word in their native language which they have never encountered before. In essence, the gap occurs when the agent is unable to do anything with the given term because they do not understand it. For example, an agent trained to recognize and handle terms that have to do with cooking would likely encounter a Lexical Gap if asked to handle something regarding a rocket. There has been a good bit of prior research on the topic of lexical gaps; for example, research was found endeavoring to match unknown terms to known terms by the use of translation models [28], as well as by adding ontology labels to natural language

expressions to find overlap with a learned lexicon [21]. This gap can be resolved rather simply by educating the agent in question on the meaning of a new word and linking it contextually to other words.

4.1.1.1.2 Homonym Gap Some words have multiple meanings depending on their context - sometimes this is due to a colloquialism which changes the meaning of a word, or sometimes it is simply that a word has different meanings depending on context. When a recipient is unclear about which sense of the word is intended, this causes a Homonym Gap. For example, if the agent is asked "Is this right?", it may be unclear whether "right" in this context means "correct" or the direction opposite of left. Prior work was found in differentiating homonyms which works by manually classifying the meanings of a sample set of instances of a term and using a neural network to classify other occurrences of the homonymous term [36, 41]. One way to resolve this gap is to look at context to resolve the intended meaning of the word.

4.1.1.1.3 Part-Of-Speech Gap A Part-Of-Speech Gap is a fairly straightforward concept and is somewhat related to the Homonym Gap. This type of gap occurs when it is unclear which part of speech a word given to the agent is supposed to be. This can arise due to the word being a homonym with a different part of speech, but could also occur when there is an unspoken subject in the phrase, which could lead to reference issues by the agent. As an example, consider the phrases "Target the letter" vs. "The letter is a target"; the word "target" serves as a different part of speech in each case. This is a well-documented problem which has been the subject of much work - notably, Eric Brill's 1992 work on an automated part-of-speech tagger [9], which has spawned a number of subsequent works in the years since, with more advanced and accurate methods arising over time. Similarly to the Homonym Gap, a way to resolve this is to pay attention to context and try to determine the meaning based on the structure of the rest of the sentence.

4.1.1.2 Negation Gap

A Negation Gap occurs when an agent understands a term but does not directly understand its negation. For example, with a Negation Gap present, if the agent is told that "the task is inactive" and is then asked "is the task active", the agent would not know the answer, rather than knowing that the answer is no. There is some prior research on detecting negations in medical writing [20, 23], as medical writing has a somewhat more constrained grammar; [20] takes the approach of finding negated phrases and determining what is negated, while the approach detailed in [23] involves constructing a parse tree from the sentence and determining the negation from there. [13] also briefly addresses this concept and discusses how the authors dealt with it. Seemingly, the simplest way to deal with a negation gap is to detect whether there is a negation present or not in a statement or query - if there is a negation, then steps must be taken to resolve what the negation means and then connect it to the existing knowledge.

4.1.1.3 Sentiment Gap

The type of knowledge gap which occurs when the recipient of some statement fails to understand the sentiment behind the statement or information is labeled as a Sentiment Gap. This occurs in situations such as when somebody misses the sarcasm in something or is unsure of whether somebody is truly upset or merely joking. Text interactions are especially prone to causing Sentiment Gaps. For example, consider the phrase "Yeah, right." At face value, it seems to be a positive statement affirming the correctness of something. However, when spoken sarcastically, it means quite the opposite. This type of gap has been the subject of much work, as sentiment analysis has been a popular field of research recently. [34] gives a general survey of sentiment analysis, its value and implications, and some approaches, while [1] gives an example of sentiment analysis on Twitter data. The research on sentiment analysis has been ongoing for quite some time and these give good milestones to demonstrate how this gap is quickly becoming more and more easily resolved. One of

the simplest methods for an agent to resolve a Sentiment Gap without these techniques, however, is to simply ask for clarification about the intended sentiment behind a statement.

4.1.1.4 Target Gap

A Target Gap occurs when it is unclear what the target of an instruction or phrase is supposed to be; this is not quite the same as considering the subject of the phrase, as the target may be implied or different from the subject. For example, consider the phrase "I expect success". Does this mean that the speaker expects the listener to succeed, that the speaker expects some form of success for themselves, or that the speaker expects some third party to succeed? Another way this gap could manifest itself would be if there were multiple identical items and a question were asked about them. For instance, consider an experiment with three red squares on a screen and the subject is told to "click the red square". Naturally, this would lead to confusion. No existing research was found addressing this problem, but it may exist under field-specific keywords. A potential resolution for this could consist of parsing the sentence and finding any sort of ambiguity as to who the target would be (whether due to having to make an assumption or simply not being able to fill in what the target is) then seeking clarification or using some confidence-based model to guess.

4.1.1.5 Open Statement Gap

An Open Statement Gap occurs when a statement is given that could cause a recipient with a closed-world perspective to make an incorrect inference. For example, take a system that only knows about the United States and Canada and receives the statement "Paris is a city which is not in the United States"; it is very possible that the system would assume that, since the only other country it knows of is Canada, Paris is a city in Canada. This is directly addressed in [10], as well as [13] - in both cases, an automated system is discussed, as well as how it deals with such a gap. An important step to resolving this problem is to have

knowledge about whether an open or closed world is currently being discussed. Once the type of system is known, different rulesets can be considered to handle queries and new information.

4.1.1.6 Context Gap

A Context Gap occurs when seemingly disjoint information is given and an agent does not know what to do with it; it can also arise if multiple phrases share the same subject but this is not immediately obvious. For example, consider the following information: "There is a target. The letter is red". Without outside context connecting the fact that the target in question is a letter, an agent would be incapable of doing anything with this knowledge. This example also shows how context can be critical to connecting two different terms for the same subject. [35] tangentially touches upon this topic, with more focus on a single sentence with complex interactions between entities, but this still works to address the confusion around subjects. The simplest resolution for this is to note that there are two seemingly disjoint entities with no prior or further reference and query if there is a connection between them or if they are intended to be separate.

4.1.2 Spatial Gaps

Spatial Gaps are a type of knowledge gap which were briefly considered as an exercise in completion; essentially, a Spatial Gap occurs when someone is disoriented or misjudges something in the spatial world around them.

4.1.2.1 Direction Gap

A Direction Gap is defined as the error that occurs when somebody gets disoriented and is unsure which direction is north or south, or which way is up or down. One example is when someone spins around with their eyes closed and then opens them - for a brief period

of time, they are unsure which way they are pointing or which way something is. Alternatively, another example would be stepping out of a car after having slept as a passenger - one must reorient themselves to figure out where they are. [11] surveys how different beings re-orient themselves using the geometry of the space that they are in, while [27] considers how geometric and non-geometric cues are used to re-orient in a space. Judging by the relevant citations, it appears that the natural way to reorient oneself is to understand the geometry of the space one is in, then from there identify recognizable traits and landmarks to understand one's position within the space.

4.1.2.2 Size Gap

A Size Gap occurs when somebody over- or underestimates the size of an object or of a distance. As an example, somebody may be mistaken and believe that Pennsylvania and Ohio are much further apart than they truly are, or that the distance between two houses is much less than a quarter mile, even when this is not the case. [8] considers the relationships between the distance, shape, and size of the object and how humans perceive each of these things and the interactions between these perceptions, while [39] analyzes how retrieving information from memory leads to underestimations of sizes and distances. There is no cut-and-dry way to correct such misinterpretation of space, other than simply measuring or testing the size of something - however, having a known object to use for scale could also be very helpful in addressing this error.

4.1.3 Reasoning Gaps

Reasoning Gaps are the term used for knowledge gaps which arise when attempting to cognitively reason through a problem and finding difficulty in the process. These can appear in many forms, but the base cause is typically a lack of knowledge about some instruction or relevant information associated with the task to be done.

4.1.3.1 Replication Gap

A Replication Gap arises when someone has demonstrated how to perform a task, but when someone who watched the demonstration attempts to perform the task themselves, they find themselves stuck at a certain point and unable to recall what needs to be done. This is functionally the same as not having succeeded in fully learning the task. As an example, consider a student taking a calculus course who is shown how to resolve a problem with partial fractions, but then goes home and finds himself unable to complete the homework. [46] proposes that some tasks are difficult to learn due to cognitive load caused by the complexity of the task - extrapolating from this, it can be inferred that a Replication Gap could be in part caused by too great a cognitive load due to too many new concepts thrown in at once. A simple way to resolve this, if possible, would be to break down the task into smaller sub-tasks and instruct each of those at once. However, as addressed in [46], some tasks simply cannot be broken into smaller portions as they are interconnected.

4.1.3.2 Procedural Gap

A Procedural Gap is similar to a Replication Gap, but differs in that it does not involve a demonstration; rather, a Procedural Gap occurs when a person knows how to do something theoretically, but when it comes time to perform the task, despite knowing all of the instructions and understanding them, the person is not able to successfully complete the task. As an example, someone may know how the steps to tie a knot, but when it comes time for them to do it, they forget how or miss a step and get the knot wrong. No prior research was found on this - perhaps due to the fact that this is a fairly similar concept to the Replication Gap, so the research on this topic is conflated with the previous gap. While resolution would be difficult, one common adage which may hold true in this case is that "practice makes perfect" - attempting to perform the task, finding the points where failure occurs, and attempting to resolve those difficulties, whether by trying alternative methods or studying the process and learning the correct method that way.

4.1.3.3 Cue Gap

A Cue Gap is based on the concept of "cues" and their validities for decision making, as explained in Gigerenzer and Todd's book on simple heuristics [19] - in a Cue Gap, a person trying to make a decision based on heuristics (consciously or not) is attempting to use cues about their options, but is lacking information on some of the cues or their validities. This can actually be positive and lead to better decisions being made, as outlined in the book's conclusions on Less-is-More decision making, but it is still a reasoning gap. An example, as explained in [19], would be someone attempting to resolve which of two foreign cities is larger, knowing some basic information about each city (whether it is a state capital, etc.). The gap arises when there are certain cues (e.g., whether the city has a soccer stadium) which are present for one city but not the other, so an adequate comparison cannot be made. The simplest way to resolve this gap is to query for additional information about the lacking cues; alternatively, one could eliminate those cues which are incomplete from consideration.

4.1.3.4 Boundary Gap

A Boundary Gap is a rather metacognitive form of knowledge gap, but as it is closely tied to reasoning, it is considered as being in the category of Reasoning Gaps. A Boundary Gap occurs when someone attempts to perform a task that is beyond their current capacity - namely, not knowing the bounds of one's own reasoning capacity, or when they expect someone else to be able to perform a task beyond the other person's capacity - in other words, not knowing the bounds of another's capacity. [6] analyzes the way in which children believe that everybody else knows what they themselves know, as well as how they do not recall false beliefs that they previously had which have since been corrected; this illustrates quite well the way in which one would have a gap in comprehension of their own past boundaries, as well as others' current boundaries. Similarly, [24] studies the way in which one's own knowledge can taint one's cognition of another person's knowledge, leading to

hesitation or errors in reasoning despite "knowing" what the correct choice would be. The brute-force resolution for such a gap would be to continue attempting new, more difficult tasks until one reaches the point that they are no longer capable, at which point the current boundary has been found; this is, of course, an over-simplification.

4.1.4 Philosophical Gaps

Philosophical Gaps are likely the most nebulous of the categories of knowledge gaps - at their core, they are gaps which have to do more theoretical cognitive and metacognitive processes which are in the realm of philosophical debate and consideration.

4.1.4.1 Explanatory Gap

An Explanatory Gap is a concept put forth by Joseph Levin in [29]. The gap occurs when someone understands what something is, but does not understand what it does or how it feels. An example based on [29] would be explaining to a computer system that pain is the firing of C fibers - the system would still not understand what pain feels like. [40] also proposes the concept of a computational explanatory gap, which is similar to the philosophical explanatory gap. Where the traditional explanatory gap consists of trouble mapping concepts to understanding, the computational explanatory gap consists of difficulty in mapping computational processes and algorithms to low-level circuitry and neural processes. There is no easy way to resolve this gap, as something which can only be felt by experiencing it is nearly impossible to synthesize the feeling of.

4.1.4.2 Update Gap

An Update Gap is also inspired by studying Gigerenzer and Todd's book [19] - in the book, there is reference to the theory that information is stored and updated in a person's memory as new and relevant data is introduced, which will lead to changing the stored informa-

tion and forgetting the out-of-date information. An Update Gap occurs when someone has failed to update stored information based on current data. This is commonly known as an error of commission when it comes to one's memory, as it is a gap caused by wholly believing something which is not the case. As an example, despite knowing that a childhood friend is 25 years old, if someone has not seen that friend often in recent years, they may unconsciously think of their friend as still being younger, as they had seen each other far more often at that age and thus have a stronger mental connection between them and that age. This is in line with [7], which holds that memory updating does not work as well when the context of the recollection matches more closely to the context in which the out-of-date information was stored. The most straightforward way to resolve an update gap is to attempt to repeatedly expose oneself to the updated information and try to create a stronger connection to the new information than to the old information.

4.1.4.3 Memory Gap

While the Update Gap is caused by errors of commission, the Memory Gap is representative of the other type of memory error: an error of omission, which consists of forgetting something entirely. As an example, an error of omission would be encountering an old acquaintance and completely forgetting their name. [5] suggests that such forgetfulness could actually be beneficial in some respects, which is supported by the aforementioned Less-is-More heuristic in [19], which holds that in some cases, having less knowledge actually leads to better results in decision-making and heuristic usage than having 100% of the relevant knowledge. While this gap can be positive in some cases, it is still important to recognize it as a gap to know how best to harness its benefits, as well as when to simply try to correct it. The clearest path to resolving a Memory Gap is to recognize which information is forgotten and to query the relevant resources or other agents to acquire the missing knowledge.

4.1.4.4 Metonymy Gap

When someone conflates a part of a whole and the whole itself, this is deemed a Metonymy Gap. This is certainly a gap likely to occur within an artificially intelligent system, as in the case where an item is misinterpreted to be the target of a search rather than recognized as a part of the target, but it can also occur with natural intelligence - for example in the case of somebody asking a question of a college freshman and assuming that their response mirrors that of the entire student body. [14] discusses the different forms that metonymy can take on and gives an analysis of the different conceptual mappings. [44] and [43] present an interesting way of classifying interpretations of metonymic phrases, which could be beneficial to helping an artificial agent understand the meaning of a metonymic idiom. Resolving this gap is difficult, as it consists of needing to know when a metonymic mapping is correct or incorrect and when to accept it.

4.1.4.5 Rationality Gap

A Rationality Gap is a conceptual knowledge gap that is related to game theory. If someone is in a situation where there is some decision to be made where there is a clearly rational choice, if the person does not know what the rational choice would be or simply chooses to make an irrational decision, this is called a Rationality Gap. A simple example is that of a Nash Equilibrium - if one of the players chooses an option in the game which would not cause the equilibrium, this is an irrational choice and so would consist of a Rationality Gap. [12] suggests that humans, while having a tendency to make errors and be irrational at times, will overall act rationally; thus, the cases where a Rationality Gap would likely arise would either be in a human acting irrationally and so erring from normal behavior, or potentially in an artificial system developed erroneously or irrationally and so into which irrationality is programmed. There is no direct simple resolution, apart from correcting course by demonstrating why a behavior is irrational and why the rational behavior would be more beneficial.

4.1.4.6 Ignorance Gap

One of the hardest gaps to recognize and resolve, an Ignorance Gap is when someone does not know what they do not know. Often, when studying or researching, it becomes necessary to build knowledge that one does not have, but this can be very difficult to do as the person does not know what they do not know until they know it. [25] analyzes how people can relatively quickly return whether they know that something is not the case using fairly closed-world examples. Meanwhile, [4] provides representations for how the mind handles uncertainty in a more mechanical sense. In a closed world, it is not too difficult to detect whether some knowledge is had, as it is possible to simply subtract current knowledge from the complete world and find the difference; however, in an open world, there are significantly greater challenges to finding out what is not known. The typical way in which one grows their knowledge in the real world is to be instructed by somebody with more knowledge, or to stumble upon some new trail of information which leads them to something they had never learned or considered before.

4.2 KOIOS

A system developed using the Python programming language was built to handle ACE instructions as input instructions into a network, which can then be queried via user questions. The system then queries the network constructed from the instructions to answer the queries with either a "Yes", "No", or "Unknown" answer. If a knowledge gap is encountered during the process of answering the query, the system identifies the gap, informs the user, and resolves the gap. This behavior can be controlled through a control panel which allows users to individually enable or disable either or both of gap identification and automated resolution for each gap specifically. This system is called KOIOS [1].

[1]The code for KOIOS is made available at https://github.com/schmidtDTN/KOIOS

4.2.1 ACE/DRS language structure and APE

The input instructions given to KOIOS are in a format known as Discourse Representation Structures (DRS) [17]. DRS is produced from Attempto Controlled English (ACE) using the web service interface of a system known as the Attempto Parsing Engine (APE), all of which is described in [16]. ACE instructions were generated for the Psychomotor-Vigilance and Visual Search tasks, then run through APE to have DRS instructions which can be fed into KOIOS.

4.2.2 Input Processing

The DRS instructions given to KOIOS are passed in through a text file, at which point the system processes each instruction depending on whether it is an instruction or part of a conditional. As the file is processed, a node-edge network is constructed to be used for the querying and knowledge gap handling. The differentiation between instructions and conditionals is used when constructing the graph. Instructions are used to directly create nodes and edges describing statements about the task. Conditionals also create nodes but those are identified by edges which connect them to instructions that must be true to trigger the consequence of the conditional, as well as to nodes that are triggered as a consequence of the conditional.

Once the file is fully processed, questions can be asked by the user by typing questions in ACE format into the terminal. While the eventual goal is to make KOIOS fully generalizable with regards to DRS input, the initial proof-of-concept implementation described here includes only a subset of DRS structures. Future expansion to include all DRS structures is a straightforward task.

4.2.2.1 Instructions

Instructions, as stated above, are lines in the incoming DRS file which are statements and are accepted as true. These instructions go through an instruction switcher which handles the DRS statement and either inserts nodes or appends to existing nodes in the graph, based on the details of the predicate and node reference given in the DRS.

4.2.2.2 Conditionals

Conditionals are identified and flagged in KOIOS by tracking the headers in the DRS to identify which lines are the condition and which are the consequence. The conditionals are processed after the instructions - each instruction within a conditional is handled in the same way as a normal instruction. However, at the very end of the conditional, a "Conditional" node is created; each node created as a part of the condition is connected to this central node by a "Condition" edge, while each node from the consequence is connected to it via a "Consequence" edge. This allows simple identification of which nodes were created as part of the normal instructions as opposed to as part of a conditional.

As an example, two graphs are provided below. Figure 4.2 shows the graph for the instruction set "There is a task named Psychomotor-Vigilance. The task is active. There is a button named 'Acknowledge'." In this graph, there are three distinct node clusters (task, button, and active), with the "task" node and the "active" node joined by the "be" node, which defines their "is" relationship. In Figure 4.3, a similar set of instructions is used, but with a conditional. This instruction set is "There is a task named Psychomotor-Vigilance. If the task is active, there is a button named 'Acknowledge'." In this graph, there are the same three node clusters with the "be" node joining the "task" and "active" nodes. However, a conditional node is introduced, which shows the "be" and "active" nodes as conditions, and the existence of the button as the consequence.

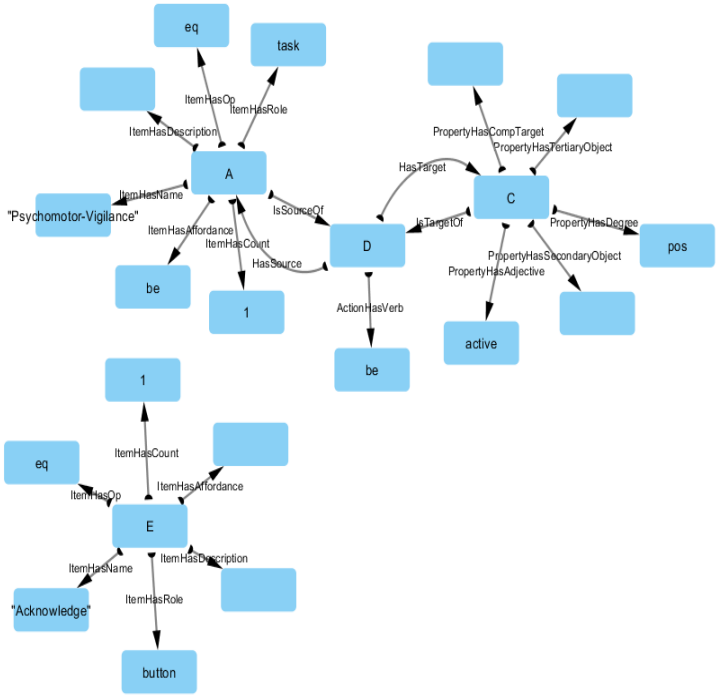

Figure 4.2: Network With No Conditional.

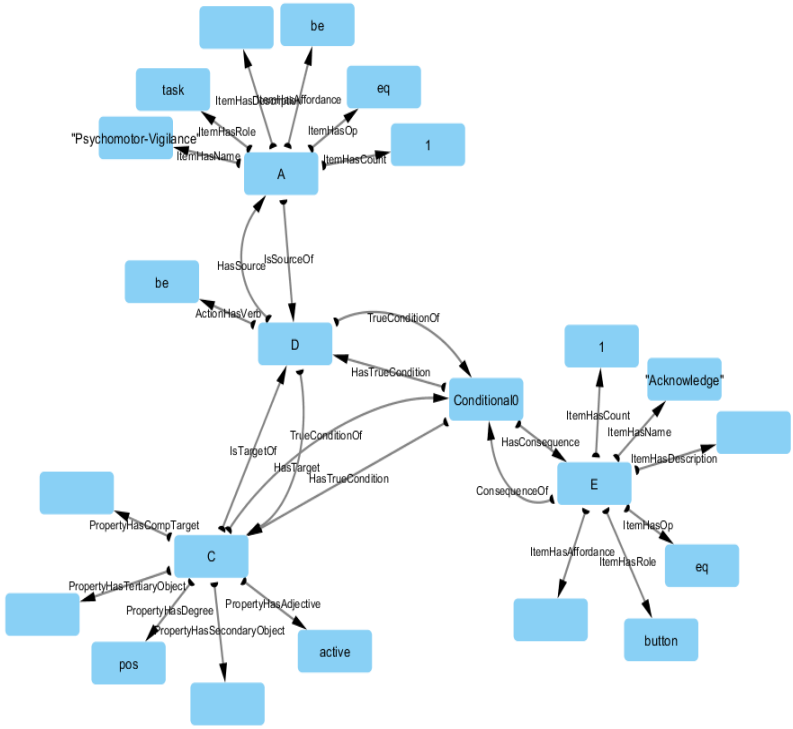

Figure 4.3: Network With Conditional.

One question which remains is whether the conditional instructions should be handled in the same way as normal instructions when it comes to queries; for example, should a question along the lines of "Does this event occur?" have within its scope the conditional nodes, or should it only focus on instructions? For the purpose of this thesis, this was addressed by allowing the question to look at the conditional nodes. This is in part for simplicity's sake, due to the limited time and scope of the research, but also because, in the current scenario of the project, it makes sense to consider the graph an image of an "ideal" task, in which case all of the conditionals would trigger. However, it would be valuable to develop a way of handling the other scenario - namely, not considering conditionals unless there is a stated instruction that a conditional has been triggered. As an example, consider the conditional "If the task is active, press the button". In the current state, a query about whether the task is active would return true, as the condition would be considered to have been triggered and thus valid knowledge. In a scenario where conditionals are not seen as being in their "ideal" state, the query would return false, due to not having an instruction specifically stating that the task was active.

4.2.2.3 Questions

Questions are entered into KOIOS by the user writing questions into the terminal. The question must be in ACE, as the APE Webservice is called to process the question and convert it to DRS. The DRS is then processed by KOIOS to determine if a set of properties and objects with the same values as those identified in the question exists in the network. If the items referenced exist in the graph, then the connecting edges are checked to see if there exists a relationship that mirrors what is asked in the question. If an appropriate edge showing the correct relationship is found, then the answer to the question is positive. If an edge that shows the opposite (or an edge to an opposite value) is found, a negative answer is returned. In any case where it is unclear what the relationship of the items and properties is, whether that is because of not being able to find a requested element or there being a

lack of connecting edges between them, the response states that the answer is unknown.

4.2.3 Graph structure

The structure of the node-edge network used in KOIOS is loosely based on the Instruction Ontology developed as part of the greater AFRL project with which this research was intended to be cooperating. In this network, nodes represent objects and properties created from DRS instructions and edges represent the relationships between those objects and properties. In addition, each object and property node has one or more attribute nodes connected to it which contain information regarding role, name, or such other attributes which affect the main node. Visual examples of the graph for the cases used in the evaluation are available in Appendix A (Figure A.1) and Appendix B (Figure B.1).

4.2.3.1 Instruction Ontology

The network's structure, as mentioned above, is loosely modeled after the Instruction Ontology. While the Instruction Ontology is much more detailed and handles many elements differently, because this research was occurring under the purview of the same over-arching project, it makes sense to have some sort of similarity between the network designed for this provisional system and the ontology used for the larger project. Additionally, this makes it less difficult to modify KOIOS in the future to operate with the Instruction Ontology as its framework.

4.2.4 Gap Identification and Resolution Mechanism

KOIOS is capable of automatically identifying certain knowledge gaps as well as resolving them, in some cases without requiring user intervention. This is configured through a "control panel" file which contains a number of booleans, each of which sets whether or not the system will identify and/or resolve a certain type of gap (identification and resolution

each have their own control). When the identification option for a gap is toggled on, the system alerts the user any time that gap is encountered, whether it is a problem or not; when the resolution option is turned on, the system automatically attempts to resolve the gap if possible rather than simply returning a response of "unknown".

4.3 Gaps Addressed

In this section, which gaps KOIOS is currently capable of identifying and resolving is discussed, as well as how the system performs these tasks for each gap.

4.3.1 Lexical Gap

A lexical gap is identified by verifying if a term which was used in a query is not found within the network of knowledge KOIOS has built. KOIOS identifies this gap during the step where it searches for nodes matching the term queried - if no such node is found, then a Lexical Gap arises. The automated resolution process begins by searching WordNet [32] for the missing term and returning the hypernyms, hyponyms, and derivationally related forms, then trying to find any of those in the network. Any node containing one of the newly found terms from WordNet then has the originally unknown term appended to it so that the next time said term appears, there is no need to go through this process. If no nodes are found to contain any of the new terms, then the system has to ask the user to manually enter another term to check and the cycle continues.

A few weaknesses in the system which still need to be addressed include resolving the problem of concept drift, as well as handling Homonym Gaps. Concept drift can occur as new terms are appended to the existing nodes, leading to an expansion in possible matches; this could lead to highly irrelevant terms matching a node due to a trail of previous matches leading to this drift. One potential way this could be addressed would be to include a

maximum distance setting which can be configured; in this scenario, an exact match would have a distance of 0, while a term found from WordNet which matches the original term would carry a distance of 1. If a new term were found which matched the 1-distance term, then that new term would carry a distance of 2. In this way, it would be possible to limit the amount of drift by only allowing a term within some pre-selected degree of separation from the original word to be used. The Homonym Gap problem is also closely tied to this, as searching WordNet for terms related to words with homonyms (e.g., the word "right") could match with several different existing nodes (in this example "correct" or "interest", as in the case of having the "right" to something). Resolving how to deal with Homonym Gaps would likely diminish this problem.

4.3.2 Negation Gap

Negation gaps are identified and handled similarly to lexical gaps; in fact, a negation gap by nature includes a lexical gap. First, a lexical gap is identified in that a term queried for is not found in the system's nodes. At this point, the system continues to behave just as if a lexical gap were found, searching WordNet to find related terms to the unknown word; however, if negation gap identification is enabled, KOIOS also queries WordNet for antonyms, not just positively related terms. If antonyms are found, then KOIOS checks if any of those antonyms are found in the nodes of its knowledge base. If an antonym is found, then a negation gap is identified and resolved; the unknown word passed in through the query is then considered to be negatively related to the rest of the query.

4.3.3 Target Gap

Target gaps are identified when a query is passed in to the system - specifically, if a user requests information in a way that leads to ambiguity as to which of several similar or identical nodes is intended, a target gap is raised. KOIOS identifies this gap during the step

where it is processing the meaning of a query - as it searches for an item referenced in the question, if it finds multiple nodes which fit the description derived from the query, then it raises a target gap. For the time being, the way that KOIOS resolves the gap is simply by providing the user with a list of the nodes which have been found and asking which one the user intended to be the target of the question; the user responds with the node they intend, and KOIOS proceeds to answer the question with that node as the target. Automation of this gap's resolution is something which is likely within the scope of future work.

4.3.4 Context Gap

Context gaps are addressed at a very basic level in this iteration of KOIOS. In the current proof-of-concept implementation, context gaps are identified only in their simplest form - if there is an item or a property which exists without any contextual edges (edges to non-attribute nodes). While there are certainly instances where there could be multi-node context gaps (for example, a network where there are two sets of multiple nodes without anything connecting them), for the present, only single-node context gaps are identified. Additionally, there is no attempt to resolve context gaps at this time. The point in time at which this gap is identified shows that there can be a form of "passive" gap identification which occurs after the instructions are entered but before any user questions are entered, as opposed to all of the other gaps addressed, which are only identified based on user queries.

5

Evaluation

Recall that the research hypotheses under investigation are:

- First, that having automated knowledge gap identification and resolution can provide a higher percentage of non-“Unknown” responses to queries than a system which lacks such a mechanism.
- Second, from a more qualitative perspective, that having gap identification, even if there is no automated resolution available, provides insight into where a problem arose with regards to answers which returned “Unknown”, enabling users to more quickly devise a new query or debug a problem with the knowledge base than simply receiving an error without explanation.

In this chapter, the evaluation methods are discussed, as well as how the impact of automated knowledge gap detection and resolution can be verified on user queries to a knowledge base. Section 5.1 covers the evaluation design and the specific methods used, and attempts to answer any questions regarding certain design choices. In Section 5.2, the results of the evaluation are discussed.

5.1 Evaluation Design

5.1.1 Design Overview

To evaluate the hypotheses claimed above and validate the KOIOS system, the evaluation was designed with the following steps.

- Select Instruction Sets
- Develop Test Question Battery
- Run Test Questions with Varying Controls
- Evaluate Results

5.1.2 Select Instruction Sets

To show a proof of concept, as well as a minimum level of generalizability, two instruction sets were chosen as KOIOS' testbed. For the initial simpler proof of concept case, the Psychomotor-Vigilance Task's instruction set was used, as it provided a set of simple instructions while still containing enough complexity for simple gaps to arise. To demonstrate generalizability, as well as begin working on more advanced knowledge gaps, the Visual Search Task was selected as the second instruction set; not only does it introduce some additional DRS terms, but it adds complexity and potential for new gaps to arise. The ACE and annotated DRS for these tasks, as well as a visual representation of the knowledge graphs of the instruction sets (pre-query), are provided in Appendix A for the PVT and Appendix B for the VST.

5.1.3 Develop Test Question Battery

To test the first hypothesis in a quantitative format, a small battery of questions was developed for each task which showcased the different gaps KOIOS is capable of detecting and

resolving. This set of questions queries a variety of information for the elements which are fully developed in KOIOS, with each question having the possible answers of “Yes”, “No”, and “Unknown”. The goal of this battery of questions is to cover the spectrum of gaps, asking positive and negative questions, as well as possibly ambiguous questions, all of which are tailored to KOIOS’ capacities. The limitations of KOIOS are discussed later in this section, as well as in the conclusion. Less questions were developed for the VST as, despite having many more nodes and edges, it has less variety and thus gives rise to less interesting queries; also, the main novel gap which it shows is the Context Gap, which is independent of queries. The batteries of questions developed for each instruction set are available in Appendix C.

5.1.4 Run Test Questions with Varying Controls

In developing KOIOS, a “control panel” of sorts was included, which allows the enabling or disabling of the identification and, separately, resolution of each gap individually; this means that certain gaps can be resolved while only identifying others and even ignoring another group. To test the first of the hypotheses, the battery of questions can be run in KOIOS without any gap detection, then run again with full gap detection and resolution; this should show a measurable increase in “Yes” and “No” answers. This could even be done with different combinations of gaps being ignored or handled to see which sets of gaps are the most impactful, but this is not being done at this time. The way of testing the second of the hypotheses is much “softer” - it is simply noted that, in the answers which are still “Unknown”, if any gaps arose which KOIOS is capable of detecting, there is output from the system which informs the user as to which gaps have arisen. This knowledge can inform the user’s decision on how to modify their query or verify that their knowledge base has correct and complete information.

PVT Scenario			
Question	No detection	Full Detection	Full detection and resolution
Is the task active?	Unknown	Unknown TARGET	Yes/Unknown TARGET
Is the task inactive?	Unknown	Unknown LEXICAL, NEGATION, TARGET	No/Unknown LEXICAL NEGATION TARGET
Is the task ongoing?	Unknown	Unknown LEXICAL, TARGET	Yes/Unknown LEXICAL TARGET
Is the computer active?	Unknown	Unknown LEXICAL, TARGET	Yes/Unknown LEXICAL TARGET
Is the computer ongoing?	Unknown	Unknown LEXICAL x2, TARGET	Yes/Unknown TARGET
Is Psychomotor-Vigilance active?	Unknown	Unknown TARGET	Yes/Unknown TARGET
Does the subject click the button?	Yes	Yes	Yes
Does the subject click Acknowledge?	Yes	Yes	Yes
Does the subject remember the letter?	Yes	Yes	Yes
Is there a task?	Yes	Yes	Yes
Does the target appear?	Yes	Yes	Yes
Is the target the task?	Unknown	Unknown TARGET	Unknown TARGET
Does the box remember the letter?	Unknown	Unknown	Unknown
Does the subject ask the letter?	Unknown	Unknown LEXICAL	Yes/Unknown LEXICAL

Figure 5.1: Evaluation results for PVT scenario test questions. The response is provided first, followed by a list of gaps which were detected.

VST Scenario			
Question	No detection	Full Detection	Full detection and resolution
Pre-queries:	No gap detected	Context Gap due to isolated node	Context Gap due to isolated node
Does the target match the letter?	Unknown	Unknown TARGET	Yes/Unknown TARGET
Does the letter match the target?	Unknown	Unknown TARGET	Yes/Unknown TARGET
Is the target the letter?	Unknown	Unknown TARGET x2	Yes/Unknown TARGET x2
Is there a key?	Yes	Yes TARGET	Yes TARGET
Is p:R pressed?	Unknown	Unknown TARGET	Yes/Unknown TARGET

Figure 5.2: Evaluation results for VST scenario test questions. The answer is provided first, followed by a list of gaps which were detected.

5.2 Evaluation Results

As this is a first attempt at robust identification and resolution of a large collection of knowledge gap categories, there are limited opportunities for quantitative comparison with existing works. Thus, the choice was made to report evaluation results here compared with a baseline case in which no knowledge gap detection or resolution is attempted.

Before presenting the results of the evaluation, a few points must be clarified to fully understand the results. First, because of the fact that the KOIOS system incorporates new knowledge into its knowledge base and the questions were run as a battery, not individually, a lexical gap should only arise once per new term (e.g., in the case of "ongoing" in the PVT results); after it has been handled once, the new knowledge is integrated and that term will no longer cause a gap. Additionally, a Target Gap arises in every instance where a question regarding an "Is" relationship is asked - this is because there are multiple "Is" relationships in the knowledge graph for each of the tasks. In future work, this could be circumvented by checking if the subject and object of the question are connected in only one of the "Is" relationships, which would mean that is the correct relationship; however, this is not being done at this point so that gaps can be maximally detected and minimal developer biases are encoded.

In a number of instances (especially with Lexical and Target Gaps), the answer may be either Unknown or a non-Unknown based on whether the user's response was enough for the system to make a confident connection or not; in these instances, the result is labeled as Yes/Unknown or No/Unknown based on what the "optimal" answer yields vs a "non-optimal" answer. In addition, an intentional choice was made to encode a bias towards replying "Unknown" when the system cannot make a connection rather than "No". With the current state of the system, to receive a "No" answer, the system must be able to make a solid connection to a negative; another approach would be to consider a lack of connection as "No", but the choice was made to treat a lack of connection as not being able to give a concrete answer. Finally, in every case where a Negation Gap arises, a Lexical Gap arises.

By nature, a Negation Gap only arises when a term is not found but its antonym is; since the term cannot be found, a Lexical Gap is detected as well.

5.2.1 Hypothesis 1 - Success Rate

To test the first hypothesis, the battery of questions was run in KOIOS for both the PVT and VST scenarios under three sets of controls: no gap detection, detection of all gaps, and detection and resolution of all gaps. The answers to the questions were then tracked, as well as which gaps were detected for each question; a concrete answer (Yes/No) is considered to be a "success", and an answer of Unknown is considered a "failure".

As mentioned above, in instances where a Target Gap or a Lexical Gap arises, the answer may be either Unknown or Yes/No, depending on whether an "ideal" user response was given; an ideal response would lead to a concrete answer rather than Unknown. In the evaluation, the ideal answer sometimes does not make sense (e.g., resolving a Lexical Gap for "computer" as meaning "task") - this case was included, however, to prove the point that multiple answers may be returned based on user input.

As is shown in the evaluation, there is no increase in successful answers if gap detection is enabled without resolution. However, as can be seen, there is an observable increase in success rate if full gap detection and resolution is enabled; with all gaps fully handled, the success rate climbs from 5 out of 14 with no resolution to 12 out of 14 with resolution and ideal user responses in the PVT scenario. Without ideal responses, the success rate is 5 out of 14, the same as without gap resolution, largely due to the bias toward responding "Unknown" rather than "No" in a situation without a clear connection. This bias is also the reason for the Unknown response in questions 12 and 13, as these should be "No" but, in the absence of a clear negation, are defaulted to "Unknown". Similarly, in the VST scenario, the success rate rises from 1 out of 5 without gap resolution to 5 out of 5 with gap resolution. These results are summarized in Table 5.1. I believe that this increase in success rate with full resolution and ideal responses shows that the first hypothesis holds.

Scenario	Success Rate without Gap Resolution	Success Rate with Gap Resolution
PVT	5 / 14	12 / 14
VST	1 / 5	5 / 5

Table 5.1: Hypothesis 1 Evaluation Summary

5.2.2 Hypothesis 2 - Usability

As mentioned earlier, the second hypothesis can be supported with qualitative evidence based on the results of the test questions which were run. As can be seen in the evaluation results, simply having gap detection enabled gives insight as to what has caused the Unknown response in almost every case (with the exception of cases where the answer is not due to a gap but rather due to the Unknown-bias). Additionally, even without any queries being asked, the Context Gap is detected immediately on completion of instruction processing, which allows the knowledge base designer to know that there is some isolated knowledge which may need handled. Even in one case of a Yes answer ("Is there a key?" in the VST scenario), a gap is detected - this is because there are two instances of "key" in the graph, but regardless of which is chosen, the answer is true; thus, all answers are "ideal", but, despite this simple resolution, this is still a gap. Another interesting query which requires gap resolution to answer successfully is "Is the task inactive?". In this query, three gaps are raised: a lexical gap, a negation gap, and a target gap. The target gap is due to the fact that there are multiple "is" relationships in the graph. The lexical gap arises from the knowledge graph not containing the term "inactive". When KOIOS attempts to resolve this gap, it encounters a negation gap, as the graph does contain "active" and "inactive" is an antonym to "active". Because of this, the lexical and negation gap are detected and immediately resolved, while the target gap requires the user to select which "is" relationship is of interest. While the gaps raised by this query can be fairly simply resolved, KOIOS nevertheless informs the user of the gaps which were encountered, as this enables the user to understand potential flaws in the knowledge base or the query.

Because of these qualitative reasons, I believe that the second hypothesis holds as well. This hypothesis could be tested more extensively and in a more falsifiable manner in the future. One way to do this would be to provide subjects with the KOIOS system which has been trained on an instruction set and request that they query it; when a failure arises, they could attempt to resolve it using the system with or without gap detection. The subjects would be timed to see how long it takes them to resolve it, and they could also provide ratings after the fact describing ease of use and difficulty in resolution. Another potential study could be to train users on KOIOS with gap resolution enabled so that they can understand where gaps arise and what types of gap may occur, then have them use a baseline version of KOIOS without any gap identification or resolution and see if they are able to resolve gaps which arise without the help provided by the system.

6

Conclusion

Knowledge bases continue to be highly relevant and useful tools in the current technology space and will likely grow to be even more ubiquitous as systems integrate more and more data. Because of this, ensuring correctness and flexibility of these knowledge base systems is a very important goal to continue striving towards.

In the pursuit of maximizing both correctness and flexibility, having a concrete definition and understanding of knowledge gaps is critical, not only to better design the knowledge bases in the first place, but to more quickly recognize the underlying cause of errors that may arise, both in the creation and querying of a knowledge base. In this thesis, a novel development was brought forth with the development of the knowledge gap taxonomy which allows for concrete categorization of specific knowledge gaps, as well as a prototype system which allows for the automated identification and resolution of knowledge gaps with minimal user input. The intention of this research is to take both this taxonomy and the prototype system and continue their development so that they can be ported and utilized as part of the overarching AFRL project with which this research was conducted in collaboration.

Again, for the sake of convenience, the hypotheses are restated here.

- First, that having automated knowledge gap identification and resolution can provide

a higher percentage of non-"Unknown" responses to queries than a system which lacks such a mechanism.

- Second, from a more qualitative perspective, that having gap identification, even if there is no automated resolution available, provides insight into where a problem arose with regards to answers which returned "Unknown", enabling users to more quickly devise a new query or debug a problem with the knowledge base than simply receiving an error without explanation.

The results of the evaluation with regards to these hypotheses appeared to support both hypotheses, though the second was supported in a much more qualitative manner. Because of this, it seems likely that continuing research and work in developing automated knowledge gap detection and resolution would be beneficial.

Future Work

Both of the major contributions discussed in this thesis are rife with opportunities for improvement through future work.

First, it would be beneficial to continue developing the knowledge gap taxonomy such that it is comprehensive, as well as finding any additional literature to support this research. I believe that a fully developed taxonomy would be of use to the scientific community.

Second, there are many aspects of the KOIOS system which could be improved and expanded; each of these will be very briefly discussed here. From a mechanical perspective, development of KOIOS should continue to handle all possible DRS terms (specifically, questions involving "what", "why", and "how" would be very beneficial to add to the system's capabilities). Additionally, certain DRS terms such as NOT and MUST should be handled in full and prepared to handle edge cases; they, as well as conditionals, can have somewhat complex structures which are not addressed in the current version of the system. Conditionals especially may need clarification and more specific handling depending on

how the knowledge base developer believes they should be represented; either they can be represented as having occurred (a view of the instructions as being in an "ideal" state), or as needing to be triggered (a more dynamic view of the instructions). It would also be desirable to develop KOIOS to handle more gaps from the taxonomy; the structure of KOIOS is primarily focused on language gaps, which would be the simplest type of gaps to extend the system to handle. Namely, the Part-of-Speech and Homonym Gaps would be relatively straightforward to implement into KOIOS; Sentiment Gaps could be quite approachable as well. Spatial, Reasoning, and Philosophical Gaps are somewhat more complex and out of range of the current form of KOIOS; additionally, Philosophical and, to a lesser degree, Reasoning Gaps may be significantly more difficult to identify and resolve using a computational approach. Finally, the system as a whole needs general optimization and deep testing.

From a more abstract perspective of these extensions, there is potentially interesting work to be done in expanding the identification of both target gaps and context gaps, as well as automating their resolution. Additionally, as a way to prevent certain gaps from occurring, it could be interesting work to develop an understanding of when the instruction set creates duplicate items which serve no purpose and handling those preemptively. As well, the integration of new knowledge which occurs upon gap resolution could, as discussed in the Research Contributions chapter, lead to concept drift where unrelated terms become accepted as being related. Finally, there is major potential for expansion in allowing multiple instruction sets to be introduced as part of the same knowledge graph, as this would allow (with even more expansion) for existing knowledge from a previously introduced task to overlap with or be used as part of a new task introduced into the knowledge base.

Finally, from an evaluation perspective, there could be a more quantifiable evaluation of the second hypothesis; namely, that the feedback generated by the knowledge identification is useful and allows users to debug more quickly. This evaluation could take place by holding a study with subjects who are told to use the system, then tracking their response

time to successfully understand what went wrong when receiving an "Unknown" reply with no gap information and comparing that to their time when there is gap information; they could additionally give ratings about ease of use and confidence for each version as well. To bolster the first hypothesis (that queries have a higher percentage of non-Unknown answers with knowledge gap identification and resolution), users could be permitted to freely rewrite questions from the test question battery or present their own novel questions and track how the system handles these new cases. In order to confidently prove or disprove the hypothesis, this could be done with a wide enough group of subjects submitting enough questions to derive a statistically significant result about the effectiveness of KOIOS for improving the question answering rate.

Bibliography

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, Portland, Oregon, June 2011. Association for Computational Linguistics.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[4] Dominik R Bach and Raymond J Dolan. Knowing how much you don't know: a neural organization of uncertainty estimates. *Nature reviews neuroscience*, 13(8):572, 2012.

[5] Liam J Bannon. Forgetting as a feature, not a bug: the dualityof memory and implications for ubiquitous computing. *CoDesign*, 2(01):3–15, 2006.

[6] Susan AJ Birch and Paul Bloom. Understanding children's and adults' limitations in mental state reasoning. *Trends in cognitive sciences*, 8(6):255–260, 2004.

[7] Robert A Bjork. The updating of human memory. In *Psychology of learning and motivation*, volume 12, pages 235–259. Elsevier, 1978.

[8] Eli Brenner and Wim JM van Damme. Perceived distance, shape and size. *Vision research*, 39(5):975–986, 1999.

[9] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC '92, pages 152–155, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[10] Jaime R. Carbonell and Allan M. Collins. Natural semantics in artificial intelligence. *American Journal of Computational Linguistics*, September 1974. Microfiche 3.

[11] Ken Cheng and Nora S Newcombe. Is there a geometric module for spatial orientation? squaring theory and evidence. *Psychonomic bulletin & review*, 12(1):1–23, 2005.

[12] L Jonathan Cohen. Can human irrationality be experimentally demonstrated? *Behavioral and Brain Sciences*, 4(3):317–331, 1981.

[13] Allan Collins, Eleanor H Warnock, Nelleke Aiello, and Mark L Miller. Reasoning from incomplete knowledge. In *Representation and understanding*, pages 383–415. Elsevier, 1975.

[14] Francisco José Ruiz de Mendoza Ibáñez. The role of mappings and domains in understanding metonymy. *Metaphor and metonymy at the crossroads. A cognitive perspective*, pages 109–132, 2000.

[15] David F Dinges and John W Powell. Microcomputer analyses of performance on a portable, simple visual rt task during sustained operations. *Behavior research methods, instruments, & computers*, 17(6):652–655, 1985.

[16] Norbert E Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Attempto controlled english for knowledge representation. In *Reasoning Web*, pages 104–124. Springer, 2008.

[17] Norbert E Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Discourse representation structures for ace 6.7. Technical report, Department of Informatics & Institute of Computational Linguistics, University of Zurich, 2013.

[18] Dedre Gentner and Allan Collins. Studies of inference from lack of knowledge. *Memory & Cognition*, 9(4):434–443, 1981.

[19] Gerd Gigerenzer and Peter M Todd. *Simple heuristics that make us smart*. Oxford University Press, USA, 1999.

[20] Stefan Gindl. Negation detection in automated medical applications. *Vienna: Vienna University of Technology*, 2006.

[21] Sherzod Hakimov, Christina Unger, Sebastian Walter, and Philipp Cimiano. Applying semantic parsing to question answering over linked data: Addressing the lexical gap. In Chris Biemann, Siegfried Handschuh, André Freitas, Farid Meziane, and Elisabeth Métais, editors, *Natural Language Processing and Information Systems*, pages 103–109, Cham, 2015. Springer International Publishing.

[22] Matthew B Hoy. Wolfphram— alpha: a brief introduction. *Medical reference services quarterly*, 29(1):67–74, 2010.

[23] Yang Huang and Henry J. Lowe. A Novel Hybrid Approach to Automated Negation Detection in Clinical Radiology Reports. *Journal of the American Medical Informatics Association*, 14(3):304–311, 05 2007.

[24] Boaz Keysar, Shuhong Lin, and Dale J Barr. Limits on theory of mind use in adults. *Cognition*, 89(1):25–41, 2003.

[25] Paul A Kolers and Sandra R Palef. Knowing not. *Memory & Cognition*, 4(5):553–558, 1976.

[26] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[27] Amy E Learmonth, Nora S Newcombe, Natalie Sheridan, and Meredith Jones. Why size counts: Children's spatial reorientation in large and small enclosures. *Developmental Science*, 11(3):414–426, 2008.

[28] Jung-Tae Lee, Sang-Bum Kim, Young-In Song, and Hae-Chang Rim. Bridging lexical gaps between queries and questions on large online q&a collections with compact translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 410–418, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[29] Joseph Levine. Materialism and qualia: The explanatory gap. *Pacific philosophical quarterly*, 64(4):354–361, 1983.

[30] Matjaž Majnik, Matej Kristan, and Danijel Skočaj. Knowledge gap detection for interactive learning of categorical knowledge. In *Proceedings of the 18th Computer Vision Winter Workshop*, pages 94–101. Univerza v Ljubljani, 2013.

[31] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in neural information processing systems*, pages 1682–1690, 2014.

[32] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[33] Boris Motik, Bijan Parsia, and Peter Patel-Schneider. OWL 2 web ontology language structural specification and functional-style syntax (second edition). W3C recommendation, W3C, December 2012. http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[34] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[35] Martin Pickering and Guy Barry. Sentence processing without empty categories. *Language and cognitive processes*, 6(3):229–259, 1991.

[36] Fabian Pittke, Henrik Leopold, and Jan Mendling. Automatic detection and resolution of lexical ambiguity in process models. *IEEE Transactions on Software Engineering*, 41(6):526–544, June 2015.

[37] Alun D Preece and Rajjan Shinghal. Foundation and application of knowledge base verification. *International journal of intelligent Systems*, 9(8):683–701, 1994.

[38] Katherine Radeka. Three types of knowledge gaps to close, 2016.

[39] Gabriel A Radvansky, Laura A Carlson-Radvansky, and David E Irwin. Uncertainty in estimating distances from memory. *Memory & Cognition*, 23(5):596–606, 1995.

[40] James A Reggia, Derek Monner, and Jared Sylvester. The computational explanatory gap. *Journal of Consciousness Studies*, 21(9-10):153–178, 2014.

[41] Uri Roll, Ricardo A. Correia, and Oded Berger-Tal. Using machine learning to disentangle homonyms in large text corpora. *Conservation Biology*, 32(3):716–724, 2018.

[42] Jiye Shen, Eyal M Reingold, and Marc Pomplun. Distractor ratio influences patterns of eye movements during visual search. *Perception*, 29(2):241–250, 2000.

[43] Ekaterina Shutova, Jakub Kaplan, Simone Teufel, and Anna Korhonen. A computational model of logical metonymy. *ACM Transactions on Speech and Language Processing (TSLP)*, 10(3):11, 2013.

[44] Ekaterina Shutova and Simone Teufel. Logical metonymy: Discovering classes of meanings. In *Proceedings of the CogSci Workshop on Semantic Space Models*, pages 29–34. Citeseer, 2009.

[45] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. In *Journal on data semantics IV*, pages 146–171. Springer, 2005.

[46] John Sweller and Paul Chandler. Why some material is difficult to learn. *Cognition and instruction*, 12(3):185–233, 1994.

[47] Matthew M Walsh, Glenn Gunzelmann, and Hans PA Van Dongen. Computational cognitive modeling of the temporal dynamics of fatigue from sleep loss. *Psychonomic bulletin & review*, 24(6):1785–1807, 2017.

[48] Qi Wu, Peng Wang, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4622–4630, 2016.

Appendices

Appendix A: Psychomotor-Vigilance Task Instruction Set

A.1 Attempto Controlled English

Psychomotor-Vigilance is a task X1.

Acknowledge is a button X2.

X1 has a box X3 and a target X4.

X4 is a letter X5.

If X4 appears in X3 then a subject X6 clicks X2 and X6 remembers X5.

If X1 is active then X4 appears in X3.

A.2 Attempto Parsing Engine Paraphrase

There is a task X1.

Psychomotor-Vigilance is the task X1.

Acknowledge is a button X2.

The task X1 has a target X3 and a box X4.

The target X3 is a letter X5.

If the target X3 appears in the box X4 then a subject X6 remembers the letter X5 and the subject X6 clicks the button X2.

If the task X1 is active then the target X3 appears in the box X4.

A.3 Annotated DRS Output

Note: For the sake of formatting and readability, some very light modifications have been made to the DRS. Nothing which is used at any part in the KOIOS process has been changed; rather, some line numbering and indentation (which is unused by KOIOS) has been removed. Annotations have also been manually added by the author for clarity.

Table A.1: Annotated DRS for PVT.

Begin Table A.1	
DRS	Annotation
[A,B,C,D,E,F,G,H,I,J]	Defining reference variables for main instructions.
object(A,task,countable,na,eq,1)	There is exactly one task represented by A.
predicate(B,be,named(Psychomotor-Vigilance),A)	Predicate B: The task (A) has a name "Psychomotor-Vigilance".
object(C,button,countable,na,eq,1)	There is exactly one button represented by C.
predicate(D,be,named(Acknowledge),C)	Predicate D: The button (C) has a name "Acknowledge".
object(E,box,countable,na,eq,1)	There is exactly one box represented by E.
has_part(H,E)	The box (E) is a member of the group of objects represented by H.

Continuation of Table A.1	
DRS	Annotation
predicate(F,have,A,H)	Predicate F: The task (A) has a group of objects H.
object(G,target,countable,na,eq,1)	There is exactly one target represented by G.
has_part(H,G)	The target (G) is a member of the group of objects represented by H.
object(H,na,countable,na,eq,2)	There are exactly 2 objects in the group represented by H.
object(I,letter,countable,na,eq,1)	There is exactly one letter represented by I.
predicate(J,be,G,I)	Predicate J: The target (G) is the letter (I).
[K]	Defining reference variable for condition of the first conditional.
predicate(K,appear,G)	Predicate K: The target (G) appears.
modifier_pp(K,in,E)	Modifier: The predicate K (the target appears) occurs in the box (E).
=>	If the above condition (the target appears in the box) is true, then the following consequence (The subject clicks the button and remembers the letter) occurs.
[L,M,N]	Defining reference variables for consequence of the first conditional.
object(L,subject,countable,na,eq,1)	There is exactly one subject represented by L.

Continuation of Table A.1	
DRS	Annotation
predicate(M,click,L,C)	Predicate M: The subject (L) clicks on the button (C).
predicate(N,remember,L,I)	Predicate N (The subject (L) remembers the letter (I).
[O,P]	Defining reference variables for condition of the second conditional.
property(O,active,pos)	Property O: The object this property is applied to is active.
predicate(P,be,A,O)	Predicate P: The task (A) has the property O applied to it (The task is active).
=>	If the above condition (the task is active) is true, the the following consequence (The target appears in the box) occurs.
[Q]	Defining reference variable for consequence of the second conditional.
predicate(Q,appear,G)	Predicate Q: The target (G) appears.
modifier_pp(Q,in,E)	Modifier: The predicate Q (the target appears) occurs in the box (E).
End of Table	

A.4 Knowledge Graph

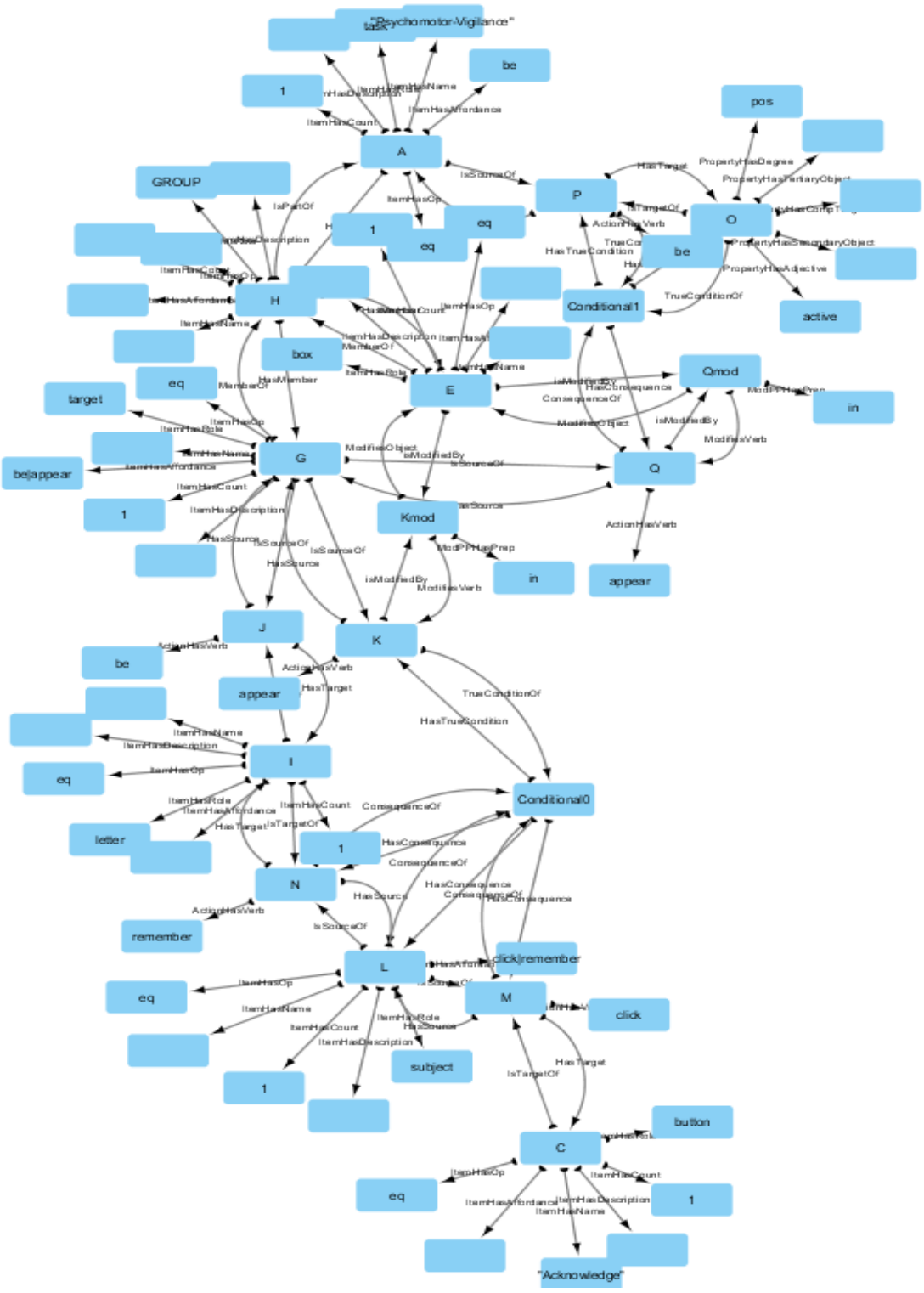

Figure A.1: Pre-Query Knowledge Graph for PVT as generated by KOIOS.

Appendix B: Visual Search Task Instruction Set

B.1 Attempto Controlled English

There are at least 2 keys.

p:W is a key X1.

p:R is a key X2.

There is a screen X3.

There is a target X4.

The target X4 is a letter.

There are at least 2 letters X5.

The letters X5 appear on the screen X3.

If the target X4 is on the screen X3 then the key X1 is pressed.

If it is false that the target X4 is on the screen X3 then the key X2 is pressed.

If there is a color X6 of the letter X8 and there is an identity X7 of the letter X8 and the color X6 is equal to a color of the target X4 and the identity X7 is equal to an identity of the target X4 then the letter X8 matches the target X4.

Every letter has a color and has an identity.

B.2 Attempto Parsing Engine Paraphrase

There are at least 2 keys.

W is a key X1.

R is a key X2.

There is a target X3.

The target X3 is a letter.

There are at least 2 letters X4.

The letters X4 appear on a screen X5.

If the target X3 is on the screen X5 then the key X1 is pressed.

If it is false that the target X3 is on the screen X5 then the key X2 is pressed.

If a color of a letter X6 is equal to a color of the target X3 and an identity of the letter X6 is equal to an identity of the target X3 then the letter X6 matches the target X3.

Every letter has a color and has an identity.

B.3 Annotated DRS Output

Note: For the sake of formatting and readability, some very light modifications have been made to the DRS. Nothing which is used at any part in the KOIOS process has been changed; rather, some line numbering and indentation (which is unused by KOIOS) has been removed. Annotations have also been manually added by the author for clarity.

Table B.1: Annotated DRS for VST.

Begin Table B.1	
DRS	Annotation
[A,B,C,D,E,F,G,H,I,J,K]	Defining reference variables for main instructions.

Continuation of Table B.1	
DRS	Annotation
object(A,key,countable,na,geq,2)	There are at least 2 keys.
object(B,key,countable,na,eq,1)	There is a key (Setup of Key #1).
predicate(C,be,named(W),B)	Key #1 is named “W”.
object(D,key,countable,na,eq,1)	There is a key (Setup of Key #2).
predicate(E,be,named(R),D)	Key #2 is named “R”.
object(F,screen,countable,na,eq,1)	There is a screen.
object(G,target,countable,na,eq,1)	There is a target.
object(H,letter,countable,na,eq,1)	There is a letter.
predicate(I,be,G,H)	The target (G) is the letter (H).
object(J,letter,countable,na,geq,2)	There are at least two letters.
predicate(K,appear,J)	The letters (J) appear.
modifier_pp(K,on,F)	The letters (J) appear on the screen (K).
[L]	Defining reference variable for condition of first conditional.
predicate(L,be,G)	The target (G) is.
modifier_pp(L,on,F)	The target (G) is] on the screen (F).
=>	If the above condition (The target is on the screen) is true, then the following consequence (The “W” key is pressed) occurs.
[M,N]	Defining reference variables for consequence of first conditional.
property(M,pressed,pos)	There is a property “pressed” (M).
predicate(N,be,B,M)	The key named “W” is pressed (M).
[]	Setting up a new conditional with no positive condition.

Continuation of Table B.1	
DRS	Annotation
NOT	The conditional has a negative condition.
[O]	Defining reference variable for condition of second conditional.
predicate(O,be,G)	The target (G) is.
modifier_pp(O,on,F)	[The target (G) is] on the screen (F).
=>	If the above condition (The target is on the screen) is false, then the following consequence (The “R” key is pressed) occurs.
[P,Q]	Defining reference variables for consequence of second conditional.
property(P,pressed,pos)	There is a property “pressed” (P).
predicate(Q,be,D,P)	The key named “R” is pressed (P).
[R,S,T,U,V,W,X,Y,Z]	Defining reference variables for condition of third conditional.
object(R,color,countable,na,eq,1)	There is a color (R).
object(S,letter,countable,na,eq,1)	There is a letter (S).
relation(R,of,S)	(R) is the color of the letter (S).
object(T,identity,countable,na,eq,1)	There is an identity (T).
relation(T,of,S)	(T) is the identity of the letter (S).
object(U,color,countable,na,eq,1)	There is a color (U).
relation(U,of,G)	(U) is the color of the target (G).
property(V,equal,pos)	There is a property “equal” (V).
predicate(W,be,R,V)	The color (R) is equal.
modifier_pp(W,to,U)	[The color (R) is equal] to the color (U).
object(X,identity,countable,na,eq,1)	There is an identity (X).

Continuation of Table B.1	
DRS	Annotation
relation(X,of,G)	(X) is the identity of the target (G).
property(Y,equal,pos)	There is a property “equal” (Y).
predicate(Z,be,T,Y)	The identity (T) is equal.
modifier_pp(Z,to,X)	[The identity (T) is equal] to the identity (X).
=>	If the above condition (Some letter has a color and identity which are identical to the color and identity of the target) is true, the the following consequence (That letter “matches” the target) occurs.
[A1]	Defining reference variable for consequence of third conditional.
predicate(A1,match,S,G)	The letter (S) matches the target (G).
[B1]	Defining reference variable for condition of fourth conditional.
object(B1,letter,countable,na,eq,1)	There is a letter.
=>	If the above condition (There is a letter) is true, then the following consequence (That letter has a color and an identity) occurs.
[C1,D1,E1,F1]	Defining reference variables for consequence of fourth conditional.
object(C1,color,countable,na,eq,1)	There is a color (C1).
predicate(D1,have,B1,C1)	The letter (B1) has the color (C1).
object(E1,identity,countable,na,eq,1)	There is an identity (E1).

Continuation of Table B.1	
DRS	Annotation
predicate(F1,have,B1,E1)	The letter (B1) has the identity (E1).
End of Table	

B.4 Knowledge Graph

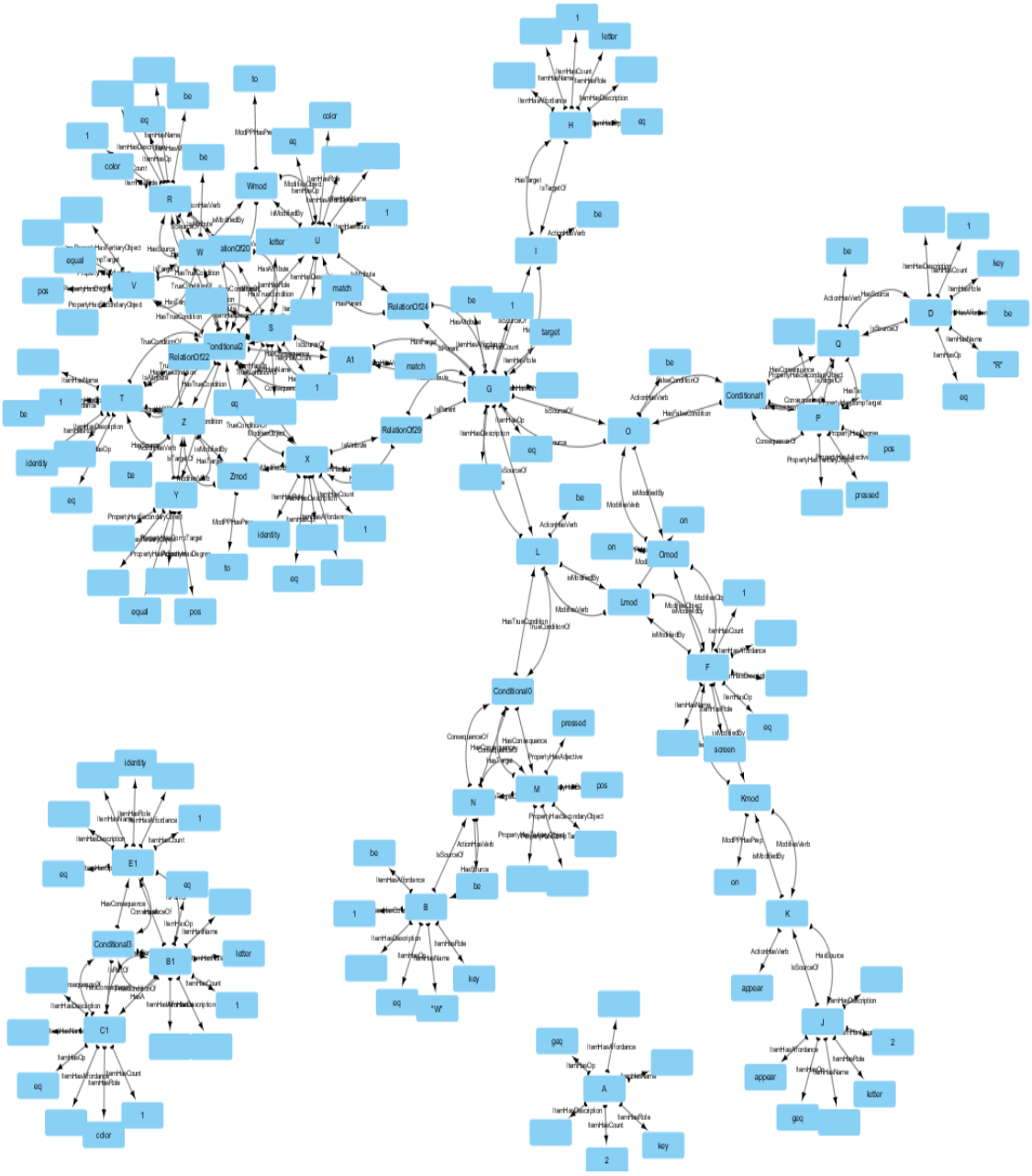

Figure B.1: Pre-Query Knowledge Graph for VST as generated by KOIOS.

Appendix C: Testing Questions

C.1 Psychomotor-Vigilance Task

- Is the task active?
- Is the task inactive?
- Is the task ongoing?
- Is the computer active?
- Is the computer ongoing?
- Is Psychomotor-Vigilance active?
- Does the subject click the button?
- Does the subject click Acknowledge?
- Does the subject remember the letter?
- Is there a task?
- Does the target appear?
- Is the target the task?
- Does the box remember the letter?
- Does the subject ask the letter?

C.2 Visual Search Task

- Does the target match the letter?
- Does the letter match the target?
- Is the target the letter?
- Is there a key?
- Is R pressed? (has to be written as “Is p:R pressed?” for the APE Webclient to properly handle).